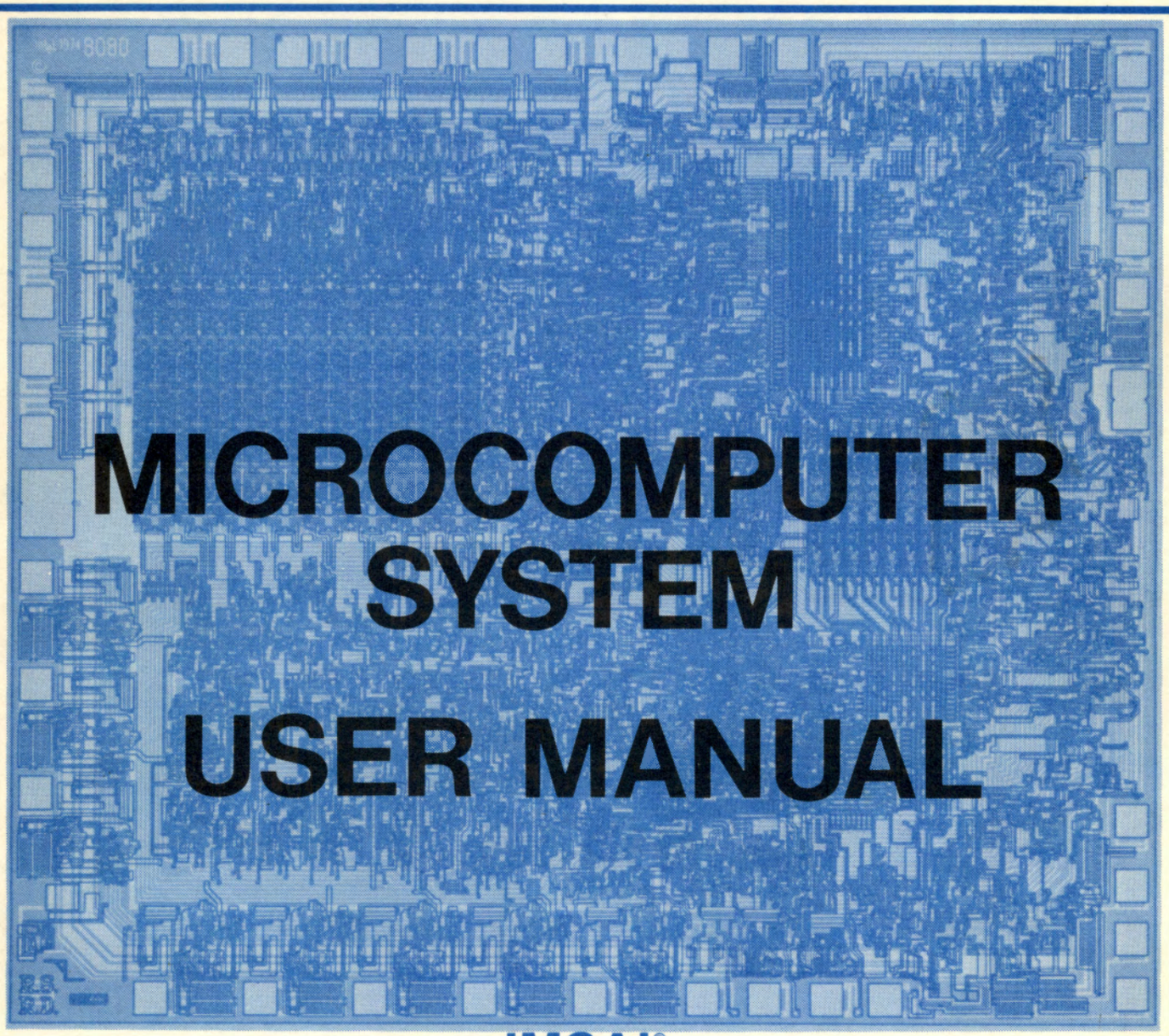
IMS ASSOCIATES, INC. USER MANUAL

IMSAI® SYSTEM



MICROCOMPUTER SYSTEM

USER MANUAL

IMSAI®

Serial NO.
004858

CUSTOMER SERVICE

REPLACEMENT PARTS

If you need a replacement part, use only standard parts from commercial sources. Use of surplus or second-run parts will void warranty. If you have trouble locating a part, write IMSAI and include:

- Part number and description as shown in the parts list.
- Serial number of cabinet or board name and revision number.
- Date of purchase.
- Nature of defect.

Note: Parts damaged through carelessness or misuse will not be replaced under warranty.

TECHNICAL CONSULTATION

Need help with your kit or system?

We encourage you to call or write IMSAI for assistance with any technical problems (except program debugging and "customizing" of hardware for special application, which we will not handle).

The effectiveness of our technical assistance depends on the information you furnish. Be sure to include:

- Serial number of cabinet and/or board name and revision number.
- Date of purchase.
- Exact description of problem.
- Everything you have done in attempting to correct the problem.
- All switch positions, connections to other equipment, system configuration, operation procedure, voltage readings and any other information that you think might be useful.

Note: Telephone traffic is lightest at midweek . . . please be sure your manual and all notes are on hand at time of call.

REPAIR SERVICE

Service facilities are available for both warranty and non-warranty repair work. If this service is desired, send IMSAI:

- Name and address.
- Date of purchase.
- Copies of all correspondence and notes relevant to the problem.
- A complete description of the problem.
- Authorization to return your kit C.O.D. for service (IF ANY) and shipping charges.
- The equipment to be repaired should be sent to IMSAI well packed.
- The original packing slip number.

Gordon

CAUTION

FAILURE TO OBSERVE THESE IMPORTANT PRECAUTIONS WILL VOID WARRANTY

1. Read all material before beginning construction.
2. Use only electronic quality rosin core solder.
3. Use extreme care with static-sensitive chips to prevent static discharge damage. (These chips are inserted in black conductive foam material in your kit.)
4. Do not plug or unplug boards while power is on.
5. Do not apply power to any board or circuit before checking each component and each trace.
6. Do not insert chips in sockets before all soldering on the board is completed.
7. Do not use nonstandard parts such as fuses of a higher current rating.
8. Do not leave out any construction step, including taping of power traces.
9. Use only specified AC power (115V AC, 60 Hz).
10. Prevent flat cable ends from touching areas of the system that may be carrying current.
11. Clean unit with soap and water or isopropyl alcohol only to prevent damage to acrylic plastic panel pieces.
12. Do not attempt repairs beyond your level of skill. Some repair operations are quite demanding to prevent damage to the board or the component.
13. Use only a 25 watt electronic soldering iron for assembly of your IMSAI 8080.
14. Do not perform any solder work on a board while power is applied.
15. Do not plug or unplug a chip from a socket while power is applied.
16. Fill out warranty cards and mail to IMSAI.

IMSAI 8080
MICROCOMPUTER SYSTEM
USER MANUAL

IMS Associates, Inc.
© 1976

Manual cover photograph of the 8080A Microprocessor Integrated Circuit Chip (Die) is reproduced with the kind permission of Intel Corporation, Santa Clara, California

IMSAI MICROCOMPUTER SYSTEM USER MANUAL

Copyright 1976 by:

IMS Associates, Inc.
14860 Wicks Boulevard
San Leandro, California 94577

TABLE OF CONTENTS

IMSAI 8080 SYSTEM
- Introduction 10
- Functional Description 12
- System Features 13
- Software Features 14
- Hardware Features 15
- Specifications 16

GENERAL ASSEMBLY AND TEST INSTRUCTIONS
- Introduction 17
- Kit Unpacking 18
- General Assembly Hints 19
- Order of Assembly 29
- Mainframe Assembly 30

CHAPTER 1 CABINET
- Assembly Instructions
- Rack Mount
- Parts List

CHAPTER 2 MOTHERBOARD
- Functional Description
- Parts List
- Assembly Instructions
- User Guide

CHAPTER 3 PSC (POWER SUPPLY)
- Functional Description
- Theory of Operation
- Assembly Drawing
- Parts List
- Assembly Instructions

CHAPTER 4 CP-A (FRONT PANEL CONTROL BOARD) REV. 4
- Functional Description
- Theory of Operation
- Assembly Drawing
- Parts List
- Assembly Instructions
- User Guide

TABLE OF CONTENTS - CONTINUED

CHAPTER 5 MPU-A REV. 4
- Functional Description
- Theory of Operation
- Assembly Drawing
- Parts List
- Assembly Instructions
- User Guide

CHAPTER 6 RAM 4A REV. 2
- Functional Description
- Theory of Operation
- Assembly Drawing
- Parts List
- Assembly Instructions
- User Guide

CHAPTER 7 PROM 4 REV. 3
- Functional Description

CHAPTER 8 PIO 4 REV. 2
- Functional Description

CHAPTER 9 SIO 2 REV. 3
- Functional Description

CHAPTER 10 PIC-8 REV. 3
- Functional Description

CHAPTER 11 UCRI REV. 2
- Functional Description

CHAPTER 12 SYSTEM SOFTWARE
- User Manual
- Listings

APPENDICES
- Schematics
  - Chassis Cabinet (Exploded View)
  - POWER SUPPLY (PSC REV. 1)
  - CP-A REV. 4
  - MPU-A REV. 4
  - RAM 4A REV. 2

Introduction

INTRODUCTION

The documentation for the IMSAI 8080 Microcomputer System consists of several books. This volume, the IMSAI 8080 Microcomputer System User Manual, contains a detailed description of the features and configuration of the computer as a complete system. It also describes the printed circuit boards that form the system building blocks. The chapter format begins with a functional description of the system or board, including brief notes about all the features. The actual operation of the system or board is then described in a theory of operations section. The physical and electronic arrangement of the system or board are next shown with a photograph and a schematic. Assembly of the board from a kit is decribed by assembly drawings or photos, a complete parts list, and assembly instructions in each chapter. Finally, the information that tells the user how to use the design features of the board to implement various functional options is contained in a user guide section for each board.

Operation of the computer as a system is documented in the IMSAI 8080 Microcomputer System User Manual in the chapter General Assembly and Test Instructions and also in the chapter on the CP-A (Front Panel Control Board). The software supplied with the basic unit, consisting of a resident monitor, assembler and text editor is described in the last chapter of the IMSAI 8080 Microcomputer System User Manual. This chapter includes both a description of the software and a complete object listing.

Supporting documentation is provided by a copy of the Intel 8080 Microcomputer System Users Manual, supplied in every system kit to give IMSAI users the primary source of detailed information about the function and instruction set of the logic implemented by the intergrated circuit chip set used in the IMSAI 8080. A handy reference card, the Intel 8080 Assembly Language Reference Card, is supplied to summarize much useful information from this manual.

To assist users in gaining a full perspective on the design and use of microprocessor-based computer systems, IMSAI includes a basic text, An Introduction to Microcomputers.

IMSAI is currently working intensively on both additional hardware (more peripheral systems, the Shared Memory Facility, etc.) and system software. Full documentation of these additions will be made available to IMSAI owners as it is produced; the charge will be a nominal fee to cover the cost of reproduction.

IMSAI 8080 MICROCOMPUTER SYSTEM FEATURES

The IMSAI 8080 is a high quality microprocessor based computer system offering outstanding capability and flexibility at low cost.

Designed to facilitate simple modular expansion, the system has both the power and the versatility to handle a broad range of data processing needs.

The IMSAI 8080 is currently supported by a broad range of peripheral devices and interfaces, and comes with a basic resident monitor, assembler, and text editor, free of charge. A broad range of high level system software is now under development, and will be available soon in both source and object form to registered IMSAI 8080 owners at the cost of reproduction and handling.

The IMSAI 8080 is available in kit or assembled form. While primarily designed as a commercial computer, the unit is configured to facilitate construction by any careful assembler. High grade industrial quality design and components are used in both kits and assembled units.

Complete documentation is provided with each system, including:

IMSAI 8080 Microcomputer System User Manual (this book).

Intel 8080 Microcomputer Systems User's Manual, completely describing the Intel integrated circuits used, and the instruction set.

An Introduction to Microcomputers, a fundamental textbook on the use of microcomputer systems.

A 90 day warranty on the system is provided. Full factory service is available at a cost commensurate with the work required.

SYSTEM FUNCTIONAL DESCRIPTION

The IMSAI 8080 Microcomputer System is a full-scale general purpose digital computer. Although small in size and low in cost, the system is exceedingly versatile and capable of data processing in the complete spectrum of practical applications.

The IMSAI 8080 uses the Intel Corporation 8080A Microprocessor LSI chip to perform the central processing function. The instruction set provided by the Intel 8080A is described fully in Chapter 4 of the Intel 8080 Microcomputer System User's Manual, provided as part of the IMSAI 8080 documentation package.

The IMSAI 8080 system is capable of unlimited expansion, due to the bus structure and IMSAI's exclusive shared memory facility, which permits parallel processing. The computing power that can be made available with the IMSAI 8080 system building blocks exceeds that of any currently available minicomputer.

The operation of the IMSAI 8080 is described in the manual chapters titled "General Assembly and Test Instructions" and in the CP-A Front Panel Control Board chapter. Input/output features are described in the I/O board chapters including SIO (Serial Input/Output board), PIO (Parallel Input/Output board) and UCRI (Cassette Recorder Interface board).

IMSAI 8080 SOFTWARE FEATURES

Basic system software (resident monitor, text editor and assembler) distributed in object form, with listing, and free of charge.

Future software releases are:

4K BASIC - upward compatible to DEC standard SUPER-BASIC

8K BASIC - upward compatible to DEC standard SUPER-BASIC

12K BASIC - DEC standard SUPER-BASIC compatible

Floppy Disk Operating System

Linkage Editor
Macro-Assembler with relocatable code generation

12K FORTRAN compiler

IMSAI 8080 HARDWARE FEATURES

Flat cable interconnection used throughout.
Absolute minimum point-to-point wiring.
Front panel has programmed output port with LED indicators.
Front panel has large easy-to-use paddle handle switches.
Front panel legends are produced photographically and mounted behind acrylic panel for protection.
Front panel has filler to increase contrast of LED indicators.
Long-life LEDs used throughout.
Front panel circuit designed so that one-shot timing links are non-critical.
No point-to-point wiring to connect or disconnect front panel to or from system.
Attractive custom designed cabinet and panel.
Rackmount cabinet available as special option.
Cabinet designed to facilitate customizing front panel.
Sturdy card cage construction.
Room for 22 cards.
Power supply subchassis with high-current transformer and computer-grade electrolytic capacitors.
Heavy duty power supply supplies power sufficient for a full complement of cards (28 amps, up to 500 watts).
Straight-through back plane wiring. No special purpose slots.
Front panel plugs into any slot to operate machine.
Double-sided printed circuit boards with plated-through holes and solder mask.
All board contact fingers are gold-plated over nickel.
PC board material is glass-fiber-reinforced epoxy laminate.
On-board power regulation. Power is regulated by integrated circuit regulators with thermal current limits.
Tantalum board decoupling capacitors. Ample .luf disk ceramic power decoupling capacitors.
Designed with latest LSI and MSI components. Package count minimized.
Heavy current tri-state bus drivers used throughout the system.
System designed from initial concept for multi-processor, shared memory options.

MICROCOMPUTER SYSTEM SPECIFICATIONS

Processor: Intel 8080A Microcomputer chip

- Directly Accessible Memory: 65,536 words
- Word Size: 1 byte (8 bits)
- Register Instruction Cycle Time: 2 microseconds
- Basic Machine Cycle Time: 0.5 microseconds
- Directly Accessible Input and Output Ports: 256
- Machine Instruction Set Size: 78 basic instructions (181 instructions with variants)
- Nested Subroutine Call Capability: Limited only by memory size
- Interrupt Capability: 8 hardware levels
- Registers: 6, plus stack pointer, program counter and accumulator

Memory Type: Semiconductor (1024x1 format chips)

Cabinet: Custom aluminum case with acrylic front panel

- Dimensions: 19½" x 17" x 7"
- Weight: 40 pounds
- Front Panel Switch Type: Paddle
- Color: IBM blue and grey

Power: 28 amp unregulated power supply with onboard regulators

- Power Requirement: Under 50 watts for basic system
- Maximum Power Capability: Up to 500 watts
- Power Type: 115 VAC, 60 Hz. single phase

Connections: Mounting space for 10 EIA-type 25-pin connectors on the back panel. Opening and cable clamp provided for flat cables to exit from the cabinet. 3M flat cable system used throughout.

General Assembly

INTRODUCTION

This chapter contains the following sections:

1. Kit Unpacking Instructions

2. Construction Hints - general notes on how to build your kit.

3. Recommended Overall Order of Assembly (includes cross-reference to chapters where specific assembly instructions for the various submodules will be found).

4. Mainframe Assembly - assembly instructions for integrating Chassis with Power Supply, Mother board and Front Panel, and instructions for testing the Power Supply.

5. System Functional Test - how to check out your overall system.

KIT UNPACKING INSTRUCTIONS

1. Remove all packages from the outer box. For a standard IMSAI 8080 kit, these will consist of:

   a. Documentation Set (Manual plus two books)
   b. Cabinet Base Plate
   c. Table Top Cover (or Rackmount cover and Rackmount painted pieces)
   d. Two large inner boxes
   e. Two small inner boxes.

2. Largest inner box contains flat parts such as pc boards, small sheet metal parts, two plastic panels and a mailing tube containing the front panel mask and paper backing sheet (latter is deleted if an OEM machine has been ordered).

3. The next smaller size inner box contains plastic sacks of components. (There will be a plastic sack with a parts list corresponding to each pc board, plus sacks for the chassis and rackmount hardware and a sack containing the paper tape for the IMSAI Self-Contained System software.

4. One of the two small boxes contains the large components for the Power Supply (transformer, capacitors, etc.).

5. The second small box is either empty (serving as a spacer box for packaging purposes) or contains overflow from the sack parts box.

6. Unpack plastic sacks only when you are ready to begin assembly of that particular module. If any parts are missing, contact IMSAI Customer Service for immediate replacement.

7. Be careful in handling the painted sheet metal parts, the plastic parts and the film negative to avoid scratching. PC boards should not be stacked without protective material between to avoid destroying or shorting traces.

CONSTRUCTION HINTS

GENERAL

The IMSAI 8080 microcomputer is a complex piece of electronic equipment. This section covers a number of items, each of which must be followed to insure a working system at the completion of assembly. This entire section must be read completely before beginning assembly, and the builder must refer back to the notes in this section often enough to insure that no components are installed incorrectly. While each assembly step is easy to do correctly, there are many steps and it is also easy to do one or more incorrectly; and much more time will be spent solving a problem than would have been needed to prevent it.

There may be items about which you are not completely sure during assembly. Should this occur, DO NOT CONTINUE. Study the manual to see if you can resolve your question, or seek the help of someone more knowledgeable in digital electronics. If you feel your question is not resolved by further study or asking whoever is available to you, call IMSAI. This will enable you to do a better construction job, and it will enable us to revise the manual so that it will be of more assistance to you. We recognize that some builders will have had very little experience in assembling electronic kits, and it is our intention to continually revise the manual based on comments by users, so that even the most inexperienced builder can achieve the best unit available with a minimum of effort. No question is too simple to call about if you're not sure about it.

TOOLS AND WORKPLACE

It is next to impossible for even an experienced builder to produce a good machine unless proper tools and an adequate workspace are available. The kit does not require much space to work in, but enough table surface should be available for the piece being worked on, all the tools needed for that piece, and an orderly arrangement of the components which will be used in assembling that piece. The work area should be very well-lit, with no shadows. If the entire room is not well-lit by ceiling or window light, then at least two bright lamps should be used, preferably one on either side and slightly behind the chair to help eliminate shadows. You may want to protect the table surface with cardboard or newspaper.

The most important single item in assembly is the soldering iron. It is critical enough that a separate part of this section is devoted to it. Other tools which are absolutely necessary to do an adequate assembly job are screwdrivers to fit the screws used in the kit (both straight slot and phillips), a small pair of diagonal cutters (preferably a 4" pair, flush-cutting), small needle-nosed pliers, and a wire stripper. A 1/4" nut driver will make cabinet assembly very much easier, as the sheet-metal screws used are designed primarily to use a nut-driver. A voltmeter should be available for testing. Any inexpensive meter (VOM) with DC voltage scales between 5 and 30 volts should do. Do not attempt to assemble the kit until you have the tools necessary; damaged parts cannot be replaced under warranty.

SOLDERING

Almost every problem with an assembled kit is a soldering problem. If you have never soldered before, or if you have done some soldering but do not yet have facility in making good soldering joints both quickly and every time, practice before beginning assembly on the IMSAI 8080 boards. Obtain some extra #20 hook-up wire and solder locally and solder pieces together until you feel comfortably able to quickly make a good joint. The importance of good solder joints is just too great to convey adequately here; but don't be scared off, because once you get the hang of it, they're very easy to do.

Soldering Irons

There are a great many tools available with the name "soldering iron". Two thirds of these are not appropriate to small electronics assembly and if used are almost certain to damage both parts and boards. The problem with most of these are that they are too big and too hot. Note that most every soldering "gun" is in the too big, too hot class. Proper soldering irons are easily available at any local hobbyist electronics outlet, and they are not expensive. Use a 30-40 watt iron with a small tip, such as an Ungar 776 with a 7155 tip. If you wish to invest in a top-quality tool, a temperature-controlled tip model such as the Weller W-TCP with a small 700°F tip is well worth the extra cost. Many irons are available with either unplated copper tips or plated tips. Though slightly more expensive, the plated tips last very much longer and give superior service.

Solder

Using the proper solder is as important as using the proper iron, and there are many solders to choose among. In normal electronics assembly, separate paste or liquid flux is not used. Rather, a solder with a "core" of rosin (or resin) base flux is used. This flux (contained in the hollow center of the solder) should be sufficient. Absolutely avoid any solders using an acid flux. (Or any cans of acid flux - unless a can of flux says"rosin" you may safely assume it is an acid flux. Acid fluxes are used for mechanical soldering where the surfaces are not as clean as those in electronic assembly. They are corrosive and will typically damage a printed circuit.

Also very important is the ratio of tin to lead used in the solder. Best to use is 63% tin, 37% lead, called 63/37 or eutectic. Much more common is 60/40, which is still a very good solder. Avoid using 50/50 or 40/60, even though they're a little cheaper. The higher-lead ratios solidify gradually, while the 63/37 solidifies almost instantaneously, making "cold solder joints" very much less likely.

Also important is the gauge (or diameter) of the solder. For fine electronics work a fine gauge should be used, such as #20 (from #19 to #22 is OK). Again, the correct solder is easy to obtain from any local hobbyist electronics outlet or TV repair shop. ERSIN Multicore or KESTER are two brands you can count on for good results. The solder included in the kit should be sufficient. If for some reason it is not, and you cannot obtain the proper solder locally, DO NOT USE any substitutes. More solder of the proper type can be obtained from IMSAI.

Soldering Technique

For a joint to solder correctly, enough heat must be applied so that both pieces of metal get hot enough to melt the solder. The tip of the iron should be applied so that it touches both the wire and the foil pad on the board. The end of the solder should then be touched to the junction of the iron, lead, and pad, so that a small amount melts and "wets" the joint (flows smoothly on both the lead and pad). As soon as the joint has wet, the iron can be removed, and the joint inspected immediately. Careful inspection of each joint is the key to successful soldering. While the solder is being applied, watch the joint carefully. You should be able to

see the solder flow onto the two surfaces. It should flow around the lead, and if you see that the solder has flowed only on one side of the lead, the iron should be re-applied (while watching the joint) to heat the joint enough for the solder to flow. (The typical reason for solder to flow only half-way around a lead is that not enough heat was applied.) For the normal joint, only a small amount of solder is needed (approximately 1/8" of 20 gauge solder wire) for it to flow all the way around the lead. Also, for the normal joint, only 2 to 4 seconds of heat applied from the iron is necessary. More heat and solder will be needed for some joints with larger leads and holes or large foil areas, but if more heat or solder is needed on typical component leads (like IC's), it is an indication that something is not right.

Since nearly all the holes in IMSAI printed circuit boards are plated-through (the inside walls of the hole have a metal surface, connecting the pads on the opposite sides and providing greater area for solder to adhere to) some solder will typically wick through and be visible on the top side of the board. This is normal. If small drops of solder appear on the top side, it is an indication that too much solder is being applied, along with more than sufficient heat. These balls of solder can easily short to neighboring pins and must be avoided. If the correct amount of heat or less than the correct amount was used along with too much solder, the solder remains on the bottom of the board (the side the solder is always applied from) and forms a blob which can easily short to neighboring pads or traces. If one of the small gaps between foil pads or traces has been shorted with too much solder, it can often be unshorted by running the hot iron lightly down the shorted trace, re-melting the solder at the shorted point and pulling it away with the iron. Do not leave iron on traces or pads too long when soldering or fixing a short, as overheated traces easily come off the board. As a result, very special care must be exercised for any component removal operation.

The tip of the iron must be kept clean to work well. Most stores that carry irons also carry small sponges in holders designed for cleaning hot tips. The tip is simply wiped on the wet sponge quickly. A damp rag will serve as well though less convenient. The tip must be kept adequately tinned at all times to avoid an oxide coating forming. It should appear bright and shiny. A small amount of solder should be melted onto the tip each time it is cleaned unless

a joint is to be made immediately. If a tip becomes oxidized, dipping it in a can of rosin flux is usually sufficient to enable solder to flow on it again. They may be cleaned of oxide by fine steel wool or other abrasive, but a plated tip should never be filed.

The tip of the iron should never have enough solder on it that it could drip off. If you find that solder tends to drip off the tip, you are undoubtedly using too much solder. A solder drip on a P.C. board is often extremely difficult to see, since it is the same color as the traces, and it is sure to short several traces and cause trouble or damage components when the board is operated. Inspect your boards very carefully for any such solder drips, shorts near soldered leads, incompletely soldered leads, and unsoldered leads. A 100% inspection of soldering should catch 99% of all problems before the board is even turned on. When soldering components with long leads (resistors, etc) we suggest clipping the leads after soldering so that lead clipping gives you an easy and positive way to check all the joints on those components. A completed unit will typically run when first turned on if the soldering was done correctly.

MOS IC HANDLING

Some of the chips in the kit are MOS type chips (such as the 8080A, 8111 and 8251). MOS chips are sensitive to static electricity and other large transient voltages. In order to prevent damaging these, some precautions should be followed. They all relate to avoiding the discharge of static through the pins on one of these chips.

Avoid working in a room with very low humidity. Wearing cotton fabric or other non-static forming fabrics will help. Air directly from a heater vent is typically extremely low in humidity and should be avoided in the work area. Keeping everything involved (chip, board, iron, tools, boxes, chip containers, work surfaces and you) at the same potential is required, and the biggest step in achieving this is continuous physical contact between them. For example, before removing a chip from a box and setting it on the table, the box should be set on the table, you should touch the table, and only then pick up the chip to place it on the table. Try to handle the chip from the ends rather than the pins as

much as possible, and always touch the chip's container or surface which it is touching before picking up the chip. Also touch a surface or container before placing the chip back in it. Touch a PC board before inserting the chip. Touch the soldering iron to the work surface or to a small piece of metal foil on the work surface before touching it to the PC board for soldering. In general, make sure the chip is not the path for any static discharge. Save MOS IC insertion as the last steps in assembly to avoid unnecessary exposure.

POLARITY

Many electronic components will not work if they are connected backwards. Any component which it is important to insert one way only will have a mark of some sort to indicate which way is which. The board where they go will have some sort of corresponding mark at each place, or an indication that all such components go the same way as a marked "typical" one.

I.C.'s

All I.C.'s must be inserted with Pin 1 in the correct location to avoid damaging the I.C. Pin 1 is indicated on the chips by several different marks. The most common is a rounded or square notch in the center of the end near Pin 1. Another common one is a slightly depressed or raised dot in the corner of the chip next to Pin 1. One or both of these will always be present to indicate Pin 1. Sometimes there are other circular markings on the centerline of the chip, usually towards one or both ends; these should be ignored. Often there is some kind of Pin 1 mark on the bottom of the chip also. (Note: Many I.C.'s have a code for date of manufacture which is a 4 digit code. e.g. 7425 would indicate manufacture in the 25th week of 1974. Do not confuse these with the device number. The code will be alone, the device number will have manufacturer-dependent suffixes and prefixes. e.g., SN7404N is a 7404 type chip. On the PC board, some Pin 1 indication will be found, such as a square pad, a dot, an arrow showing Pin 1 direction with the note "typical" (indicating all chips on the board face the same way), or similar mark.

The board or the chip is very likely to be damaged if there is a need to unsolder a chip that was soldered in with Pin 1 in the wrong direction. Unless you are completely sure you are capable of unsoldering an integrated circuit without damage to the circuit or the board, you should send the board back to the factory to have the work done for you. Remember that on these boards with plated-through holes, pins are not only soldered on the top where you see the visible bead of solder, but is soldered inside the hole which makes it much more difficult to remove.

Diodes

Diodes will typically have a band around the body, next to the cathode end. This corresponds to the bar on the typical diode symbol. The same is true for Zener diodes. A diode symbol should be found on the board or assembly diagram to indicate the proper mounting direction.

Capacitors

Some capacitors have a plus and minus lead; among them the tantalum and power supply electrolytic capacitors. Some mark on the body of the capacitor will indicate the plus lead, typically a + sign near it. There will be a mark (typically a + sign) on the board or assembly diagram to indicate the proper direction to mount the capacitors. A capacitor of this type is usually destroyed very quickly if power is applied to it in the reverse direction, so check your assembly carefully.

Transistors

Most transistors have a flat side or a small tab to indicate the lead orientation. If this indication is oriented according to the assembly diagram the leads should fit in the holes with little bending and no crossing.

MOUNTING COMPONENTS

Integrated Circuit Chips (IC's)

Some of the chips come in a little plastic rectangle with an open bottom and top. These can be used as inserters by setting the carrier with the chips on a piece of felt or similar material on a table top and pushing lightly with a pencil eraser or small object that will fit in the top of the carrier, until the chip has slid down with the leads resting against the table. Now, because of the material, the leads will be sticking out beyond the carrier a little bit. If you then pick up the carrier and the chip and set it on the board, you can line up the little protruding tips of the IC's ends into the holes into which they are supposed to go, and while you are holding the carrier, push the chip the rest of the way into the board again with a pencil eraser or with an object that will fit inside of the carrier.

For the chips that do not come in a carrier, after you insert the ones that did come in a carrier, you could use those carriers to insert the others also, by turning the carrier upside down and setting one of the other chips on the carrier and pushing it into the carrier and then just continuing the same process described above, to insert it in its location.

For chips with no such inserter aid available, the pins should be bent inwards far enough to line up with the holes in the board. Bend the pins on each side equally. The whole row of pins on one side can be bent in uniformly if they are all pressed against a flat surface to bend them. After putting the chip in the board, two diagonally opposite pins can be

bent slightly to hold the I.C. in the board while soldering.

Take special care on each and every chip to observe the following points:

1. That Pin one is in the correct direction. Refer to marking on the board or assembly instructions to determine which direction Pin one belongs.

2. After inserting the chip and before soldering, check that every pin went through the hole properly. Sometimes a pin will catch on edge of a hole and bend under the chip instead of going through. Care should be taken to avoid this happening and to check before soldering to make sure it has not happened.

After inserting one or two chips, get a feel for how much pressure is needed to push it out of the carrier. Any chips that seem to take more pressure indicate that perhaps one or more pins are not lined up with the holes properly. Most chips after insertion, will stay in the board securely due to the fact that the leads are normally bent outward somewhat and will hold the chip by pushing outward against the holes. Some chips, however, will be loose after inserted. Extra care should be taken to see that these are properly against the board when they are soldered. The board can either be set flat against the table or other surface that will hold the chips against the board or two diagonally opposite ends may be bent slightly to prevent the chip from dropping out.

Power Regulators

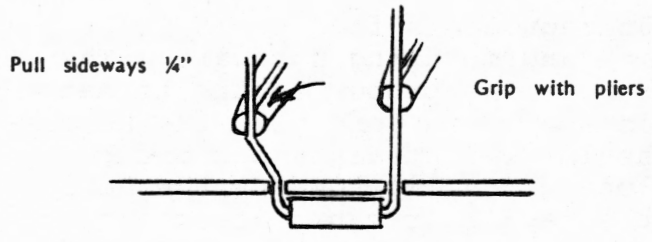
The 7805 regulators for the +5 volts are supplied with a heat sink and mounting hardware. The three leads must be bent down at the proper lengths to match the solder pads, and this should be done with the needle-nose pliers. The lead should come straight out and bend sharply down, rather than slope gradually towards the hole. After the leads are bent, the regulator can be fastened to the board along with the heat sink, using the short 6-32 screw down from the top, with the lockwasher and nut on the back. The regulator should be held to prevent turning while the nut is tightened firmly. The nut should be tight enough to insure good heat conductivity between the regulator and heat sink and board. Heat sink grease may be used if desired.

Discrete Components

Resistors and diodes can be installed most neatly using a lead bender to bend the leads consistantly. Most pads for this sort of component are .5" apart.

Disc ceramic capacitors often have the dipped insulation extending down the leads a short distance, preventing these from being pulled down all the way to the board. This insulation may be broken off by squeezing it in the pliers. Take it off until the bare wire comes up to the level of the bottom of the capacitor.

All discrete components should be held in their desired final position while being soldered. Normally this means holding them against the board by putting a slight bend in the lead behind the board so the component cannot lift from the board. (See the sketch for a way of bending the leads we find works better than simply finger-bending them slightly.) Components not held in place look sloppy and it is much harder to move them once they are soldered. In some cases, a little extra lead is needed, such as to lay the disc capacitors down on top of the chips on the front panel board. In these cases the solution is again to hold them in their final position during the soldering operation. This insures that the leads are left the proper length.



RECOMMENDED ORDER OF ASSEMBLY

Step	Description	Described In
1	Assemble MPU and RAM boards. Check carefully.	MPU Chapter RAM Chapter
2	Assemble CP-A including switches and flat cable. Check carefully.	CP-A Chapter
3	Assemble electronic components on Power Supply. Check carefully.	PS-C Chapter
4	Assemble Mother board(s). Check carefully	Mother Board Chapter
5	Assemble Chassis sheet metal: a. Install required number of card guides on card frames. b. Install fan (if supplied) on back frame. Install line cord through grommet. c. Bolt together sheet metal parts. Install rubber feet.	Cabinet Assembly Chapter
6	Install Power Supply Board in chassis. a. Bolt board in place. b. Bolt transformer in place. c. Cut wires to length and crimp on (or solder on) lugs. d. Connect up Power Supply except for wires to Mother board(s).	Mainframe Assembly Section
7	(Connect Mother boards together and) install Mother Board(s) in chassis.	Mainframe Assembly Section
8	Connect wires to Mother board. Check carefully.	Mainframe Assembly Section
9	Prepare front plastic panel assembly.	CP-A Chapter
10	Plug CP-A board into Mother board. Connect wires to CP-A board. Install front panel assembly. Hold CP-A DIP cable out of way.	Mainframe Assembly
11	Check complete assembly carefully before applying power. Plug in machine and turn on. Test Power Supply voltages.	General Assembly and Test Instructions
12	Plug in MPU board and RAM board(s) and test system.	General Assembly and Test Instructions
13	Assemble other individual boards. Check carefully.	(Individual board chapters)
14	Install individual boards.	
15	Install required cables. Install Cable Clamp.	Cabinet Assembly Section
16	Install Switch escutcheon and cover and/or Rackmount parts.	Cabinet Assembly Section

MAINFRAME ASSEMBLY

Assembly of the mainframe consists of the following steps:

- Power supply installation
- Mother board installation
- Connection between power supply and mother board
- Installation of CP-A panel.
- Connection of power supply and front panel

POWER SUPPLY INSTALLATION

Remove #8 hardware from transformer on Power Supply p.c. board. Take care to not let the transformer damage the p.c. board. Put the five #8 screws in the cabinet bottom and secure with the 8-32 threaded spacers. Install the four ¼"-20 nuts and spacers for the transformer similarly. Carefully lower the Power Supply Assembly onto the mounting screws so all the screws extend through the board. Fasten with washers and nuts. See Figure 1. Complete the power supply by attaching the capacitor brace plate to the bases of the large capacitors with the adhesive backed foam tape on one side of the brace plate.

MOTHER BOARD INSTALLATION

Attach the Mother board to the cabinet base with the hardware supplied with the Mother board as shown in Figure 2. The front 100-pin connector should be located in front of the sheet metal front frame to accomodate the CP-A assembly.

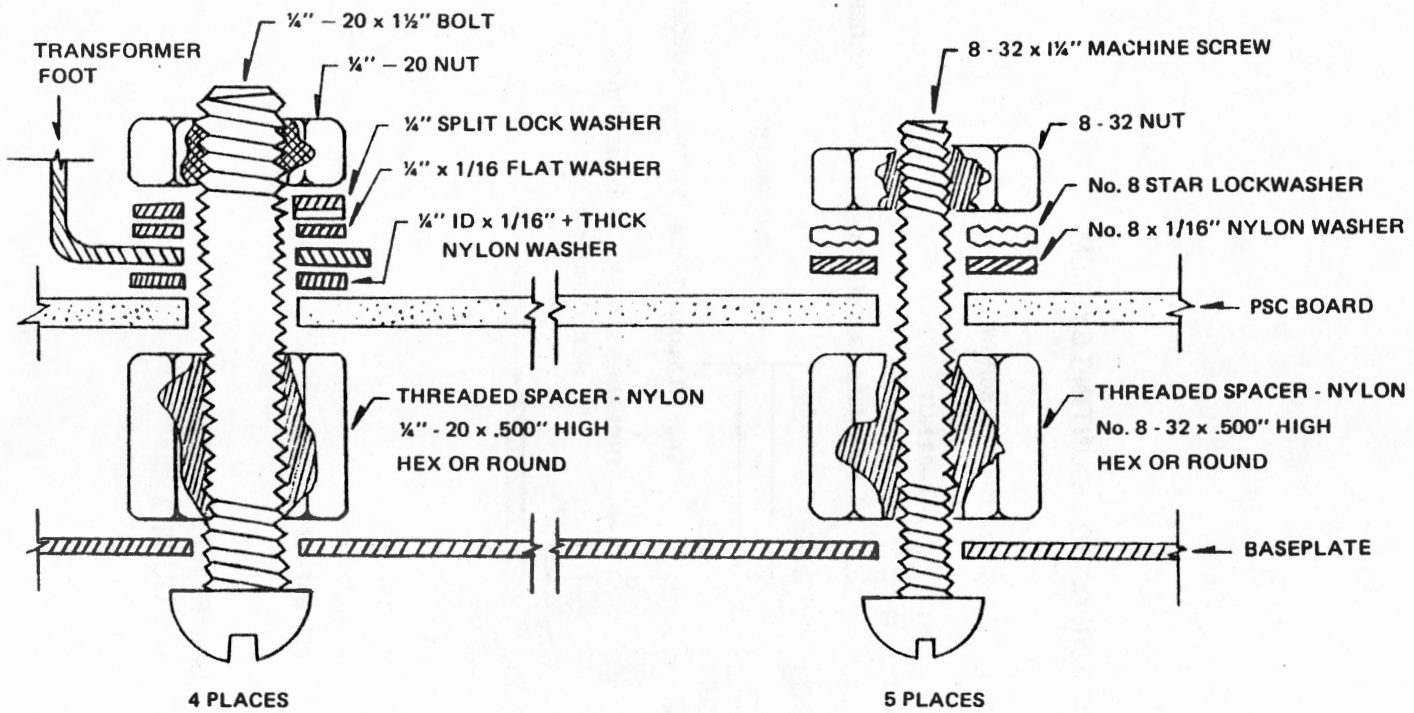
CONNECTION BETWEEN POWER SUPPLY AND MOTHER BOARD

See the wiring drawing in the Power Supply chapter. Connect the following wires between the Power Supply and the system:

a) 1 or 2 #18 gauge wire from holes at edge of -16 volt plane to -16 volt trace on Mother board.

b) 1 or 2 #18 gauge wire from holes at edge of +16 volt plane to +16 volt trace on Mother board.

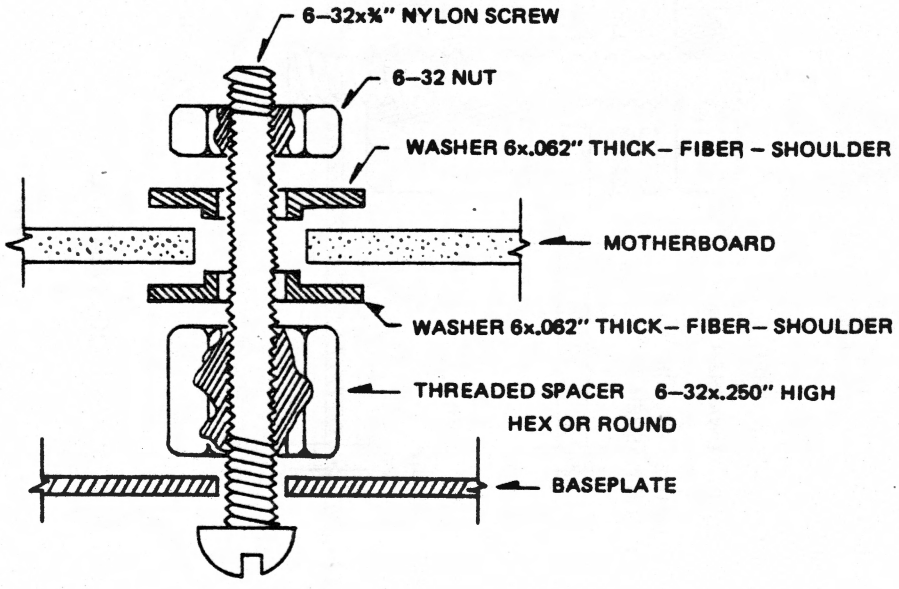
c) 2 or 3 #14 or #12 gauge wire from +8 volt plane to +8 volt bus on Mother board.



PS-C BOARD MOUNTING SYSTEMS

MAINFRAME ASSEMBLY
FIGURE 1

MOTHERBOARD MOUNTING SYSTEM



MAINFRAME ASSEMBLY
FIGURE 2

d) 2 or 3 #14 or #12 gauge wires from ground plane to ground bus on Mother board.

e) 2 #18 gauge wires from External Switch pads to power switch on CP-A or on back panel.

f) 2 wires (#18 or #20 gauge) from switched AC pads to fan (if fan installed) install insulated tubing over fan terminals.

g) 3 wires from power cord to terminals W, G, and B on PS-C. Make sure the power cord wire colors match the label on the panel

INSTALLATION OF CP-A PANEL AND CONNECTION TO POWER SUPPLY

Plug the completed CP-A panel into the front 100 pin connector on the Mother board. Install the eight Allen head screws into the PEM nuts on the sheet metal front frame. Solder the two #18 gauge wires from the External Switch pads on the Power Supply assembly to the power switch pads on the CP-A panel. Provide as much clearance as possible between the connections on the CP-A board and the sheet metal front frame. Be careful not to damage the acrylic panels with the soldering iron.

CHECK OUT OF POWER SUPPLY

Before plugging in circuit boards except the CP-A board, the unit should be plugged into the AC power supply and the power supply turned on by depressing the front panel rocker switch. The voltages at the outputs should then be measured (any DC volt meter with a full scale voltage of 20 to 50 volts will do) and the voltages should read approximately 18 volts on the +18 and -18 volt outputs, and 10 volts on the +8 volt output. If the voltage does not come to these values, a check should be made that the positive and negative terminals of the capacitors are connected properly and the diodes are mounted properly. If there is a problem with any of these items a wiring error has probably been made and the wiring should be re-checked carefully. If the wiring is checked and no error is found, assistance should be sought from a person knowledgeable in electronics or from the factory.

When the voltage of the capacitors has been checked out to be satisfactory, the unit may be turned off. A 10 minute wait will permit the capacitors to discharge. While there is considerable energy stored in the power supply filter capacitors when they are fully charged, the voltage levels are not high enough to present a danger. Some care should be taken, however, not to discharge the capacitors by shorting them with a tool or other metallic object.

With the Power Supply checked out and operating properly, the rest of the system is ready to be tested. The MPU board should be inserted in the slot behind the front panel with the flat cable inserted into the socket in the upper right hand corner of the MPU board before the board is fully seated.

The memory board should then be inserted in the third slot. While it is not necessary that the first memory board be addressed beginning at position 0, it is normally expected and the rest of this section will assume that the memory board jumpers were wired according to the directions in the User Guide section of the RAM-4A board for addressing the board at 0.

The slots in the Mother board are not unique and if a larger version (e.g., 22 slot) was ordered with more edge connectors, the boards need not be plugged into the second and third slot as directed but may be plugged into any slots.

SYSTEM FUNCTIONAL TEST

When the boards are installed, the machine is ready to test. Turn the power on with the front panel rocker switch and depress the RUN/STOP switch momentarily to STOP position and release. The WAIT light should be on and the RUN and HOLD lights should be off, with the other lights in various states at this time. Raise the RESET switch momentarily to the RESET position and release. All the lights on the bottom row in the ADDRESS BUS section should be indicating that the program counter is set to location 0. The WAIT light should still be on with the RUN and HOLD lights off. The DATA BUS lights may show various random bits on and the STATUS byte should have three lights on: MEMR, M1, and $\overline{WO}$. With all 16 ADDRESS switches in the down or 0 position, the EXAMINE/EXAMINE NEXT switch

should be raised momentarily to the EXAMINE position and released. Check that the lights after this operation are exactly the same as described for after the RESET switch was operated.

The machine is now ready to enter a small test program. For complete description of program operation in computers, read An Introduction To Microputers. For the initial machine test, the following program should be entered:

TEST PROGRAM 1

ADDRESS	HEX	BINARY	OCTAL	
0	DB	1101 1011	333	INPUT
1	FF	1111 1111	377	ADDRESS
2	D3	1101 0011	323	OUTPUT
3	FF	1111 1111	377	ADDRESS
4	C3	1100 0011	303	JUMP
5	00	0000 0000	000	LOW ADDRESS
6	00	0000 0000	000	HIGH ADDRESS

TEST PROGRAM 2

ADDRESS	HEX	BINARY	OCTAL	
0	DB	1101 1011	333	INPUT
1	FF	1111 1111	377	ADDRESS
2	2F	0010 1111	057	COMPLEMENT DATA
3	D3	1101 0011	323	OUTPUT
4	FF	1111 1111	377	ADDRESS
5	C3	1100 0011	303	JUMP
6	00	0000 0000	000	LOW ADDRESS
7	00	0000 0000	000	HIGH ADDRESS

The address is now at 0 as indicated by the lights labelled ADDRESS BUS. Into position 0 we wish to put an input instruction.

The bit pattern for the input instruction must be set in the center group of switches labelled ADDRESS-DATA. Switches 7, 6, 4, 3, 1 and 0 should be placed in the up position. Compare these switch positions with the binary representation of the input instruction listed on the first line of test program 1. We wish now to deposit this bit pattern in memory position 0. Raise the DEPOSIT/DEPOSIT NEXT switch up momentarily to the DEPOSIT position and release. The address bus should still show 0 (no lights lit) and the data bus should now show the bit pattern set in the switches (bits 7, 6, 4, 3, 1 and 0 lit and bits 5 and 2 off).

Next, the bit pattern for the address of the input port should be written in position 1. This can be done by setting all eight ADDRESS-DATA switches up, corresponding with the address listed on line 2 of Test Program One, and the DEPOSIT/DEPOSIT NEXT switch depressed momentarily to the DEPOSIT NEXT position and released.

Now the address bus light should show position 1 (address bus light 0 on and all other address bus lights off). The data bus should show all eight lights lit corresponding to the bit pattern written here. Similarly, the next five lines of Test Program One should be set into the ADDRESS-DATA switches and deposited by operating the DEPOSIT NEXT switch, each time checking to make sure that the data bus lights correspond with the settings of the ADDRESS-DATA switches and that the address is correct indicating that no steps have been skipped or done twice.

When the last byte has been deposited in address position 6, then all 16 address switches should be returned to the 0 position (down) and the EXAMINE switch operated. This should reset the address bus lights to 0, and display the contents of the bottom word in memory on the data bus lights. (This should still be the binary pattern listed in line 1 of the Test Program). The EXAMINE NEXT switch can then be operated and the address bus lights should indicate address 1 (bit 0 on and all other bits off). The Data Bus should show the contents now of memory location one which should correspond to the second line of Test Program One listing (all ones).

The EXAMINE NEXT switch can be repeatedly operated, each time checking that the data located in the consecutive memory location corresponds exactly to the listing for Test Program One.

The EXAMINE switch can again be raised momentarily with the address switches all down, to return the machine to position 0, once it has been determined that all lines listed in Test Program One are stored correctly in the memory.

Now we can single-step through this program and watch the operation of the machine. With the machine sitting at 0 with the correct instruction on the data bus, and the MEMR, M1 and $\overline{WO}$ lights lit in the status byte, the processor is reading the first instruction out of memory into the processor for execution. If the SINGLE-STEP switch is either depressed or raised once, it will permit the processor to complete its cycle and begin the next cycle. The address bus lights will show position 1, the data bus will show all ones corresponding to the bit pattern in the Test Program, and the status byte will show MEMORY READ and $\overline{WO}$. The lack of an MI light in a status byte indicates that the processor is no longer fetching an instruction to execute, but rather this cycle it is fetching the address for the instruction which it has already stored internally.

If the SINGLE-STEP switch is operated once again, the address bus lights will all be lit. The status byte will show INP and $\overline{WO}$ and the data bus will at first show no lights on. If one or more switches in the left hand group of eight switches is now raised or lowered, the corresponding light on the data bus indicators will turn on or off. The processor is now executing the first instruction which was an input data from address FF hex (377 octal) which is the address for the programmed input port on the front panel. By means of this instruction with this address the processor is able to read the position of the eight switches in the left hand group. (The address being read is indicated by the lights in the address bus and, on input or output instructions, the address appears in both groups of eight lights on the address bus. Thus, for this address, all the lights in the address bus are lit.)

The switches in the left hand group should be left in the position of some up and some down to provide a recognizable pattern before continuing. With the pattern left in the left hand group of switches, the single step switch can be operated once more permitting the processor to complete the execution of the input instruction, and begin the next cycle. Having completed the input instruction, the next cycle will be a fetch cycle during which the processor reads the next instruction to be executed, which it will find in memory address position 2. The address bus lights should now show positon 2 (bit 1 on and all others off), and the data bus should indicate the bit pattern listed on line 3 of Test Program 1 for address position 2. This is the output instruction.

The Status Byte will again have MEMR,M1, and $\overline{WO}$ lights lit and the others off. When the single step switch is operated once again, the processor is permitted to complete the cycle during which it reads in the output instruction and begin the next cycle during which it will read the address of the output device. Since it is reading this address from the next memory position, (memory position 3), the address bus will have bits one and 0 on and the others off. The Data Bus will have all lights on indicating the bit pattern we stored in memory position 3. The status bit will show MEMORY READ and WRITE OUT lights on, and the M1 light is off at this time, indicating that this is not an instruction fetch cycle, but rather it is one of the cycles required to execute the last instruction fetched-in this case, reading the address to which the data will be output. When the SINGLE STEP switch is operated once again, the processor is permitted to complete the cycle of reading the output address in and begin the next cycle which is the output operation.

The output operation looks similar to the input in that the address of the output device appears in both the upper and lower half of the Address Bus, (again in this case lighting all the lights), and the data being output appears in the Data Bus, which should show the pattern previously set in the left hand group of switches. Since the data is being output from the accumulator in the processor where it was previously stored in the input instruction, it will not be affected by moving the switches in the left hand group at this time. The Status Byte shows the MEMR light off at this time and shows the out light on indicating that the processor is executing an output instruction. The $\overline{WO}$ light is off indicating that the processor's WRITE strobe is active. If the SINGLE STEP switch is operated once more, it will permit the processor to complete the WRITE operation and begin the next cycle. At this time, the PROGRAMMED OUTPUT lights at the top left of the panel, should be lit according to the complement of the pattern that was set in the switches. That is, for each switch that was set in the up position, the light will be out, and each switch that was set in the down position, the corresponding light will be on.

Since the processor has completed the output instruction the next cycle is used to fetch the next instruction to be executed, which it will read from memory position 4. In memory position 4 we had stored the jump instruction

which should now appear on the lights on the data bus indicators. As the SINGLE STEP switch is operated again, permitting the processor to complete the fetch of the jump instruction, and start the next cycle of executing that jump instruction, we find that the processor is reading the low half of the address from memory position 5. The status byte shows the MEMR and $\overline{WO}$ lights lit, and the M1 light is off at this time.

If the SINGLE STEP switch is operated once again, it will be seen that the processor is reading the high address byte previously stored in memory location 6.

The next operation of the SINGLE STEP switch permits the processor to complete the execution of that jump instruction, which is instructing the processor to take its next instruction to be executed not from memory position 7 but from memory position 0 as was stored in the two bytes following the jump instruction.

The Address Bus lights should now be all off indicating that the processor is indeed fetching the next instruction from memory location 0. The Data Bus should show the pattern that we wrote in memory position 0 as the input instruction. We have now completed one cycle of the loop in Test Program 1. Further operations of the SINGLE STEP switch will let the processor step through the execution of the loop additional times and each time through the loop it is possible to set a different pattern in the left hand group of switches to be read in and later to be written out to the PROGRAMMED OUTPUT lights. The RUN/STOP switch can be momentarily raised to the RUN position and released. This will permit the processor to run at the full clock speed which will result in the loop being executed roughly 50,000 times every second. Thus, as any of the switches in the left hand group of eight are moved while the program is running, the machine reads the new position essentially instantly and displays it in the PROGRAMMED OUTPUT port above.

It may have been puzzling that the lights in the PROGRAMMED OUTPUT port seem to indicate the opposite of what might have been expected when a bit was read in as a 1 and output to the PROGRAMMED OUTPUT port. This will serve as an example of the way logic design has been affected by the appearance of large scale integration and micro-processors. While it would have been entirely possible

and easy to provide a circuit modification such that when the data was put out as a 1 the light would be lit rather that turned off, such as addition to the circuit would have cost you more that the cost for byte of memory. The same function as the added circuit can be accomplished by adding one instruction to the loop which complements the data, that is, changes all ones to 0's and all 0's to 1's. Test Program 2 is exactly the same as Test Program 1 with the addition of one instruction between the input instruction and the output instruction, which will complement the data read in from the switches before it is output. If the machine is stopped and reset, Test Program 2 may be entered exactly the same way as Test Program 1 was and checked and then run through one or more cycles with the operation of the machine and to double-check that the program truly has been entered correctly. Then the RUN switch may be actuated to permit the loop to run at high speed.

With this change in the program, the PROGRAMMED OUTPUT port will show a light lit when the switch is positioned up to enter a 1 bit. Not only is this a less expensive way to achieve the function of causing the lights to turn on when the bit is entered as a 1, but it is a much more versatile solution since the operator can change his mind at a later date and either remove the complement instruction or change it to yet another instruction for a different result.

When single stepping through Test Program 2, the compliment data instruction is seen to use up only one cycle of the processor. We are able to see it being fetched to be executed, and when the SINGLE STEP switch is operated again, we are immediately fetching the next instruction. This will be true of any instructions which operate only on data which is already stored within the processor. Additional cycles are only necessary if additional information must be read in or out of the program processor itself.

After either loop is running, the RUN/STOP switch may be depressed to STOP at any time and the operation processor will stop during the fetch of the next instruction. Due to the speed at which the processor operates, it is impossible to tell beforehand at what point in the loop the processor will be at the exact instant that the RUN/STOP switch is moved to STOP, so that the processor will stop at different places in the loop for different times when the switch is actuated.

The switch may be raised to the RUN position starting at any point in the loop and the processor will continue to run at high speed beginning at the point. The flip-flop set by the RUN/STOP switch simply instructs the processor to wait at each cycle for a pulse which is generated by the SINGLE STEP switch to be received before executing the next cycle, and apart from waiting for this pulse, the processor executes exactly the same whether it is in the single run mode or stop mode.

The definition of a computer involves both the ability to execute in sequence of instructions which is stored inside the machine, also the ability to make a decision between on the value of data and as a result of that decision, choose between alternate possible paths of program step sequences to execute. Test programs 1 and 2 involve only the execution of a sequence of stored program steps and do not involve any decisions. Program 3 will illustrate the use of decisions in a computer program and should provide some interesting entertainment as well. It is a game program using the INPUT switches and the PROGRAMMED OUTPUT lights on the IMSAI 8080 front panel.

A pattern of lights in the PROGRAMMED OUTPUT ports is moved to the left one bit at a time, and the left hand bit which is "pushed off" the end of the programmed I/O register re-appears at the right end of the register. The rate at which the bit pattern is shifted to the left can be chosen by the binary number set in the front panel switches when the program is first started or when the machine is reset to start again. When a higher binary number is entered in these switches and program restarted, the bit pattern will shift to the left at a higher rate of speed. Initially, switches should be set for (2), that is all switches down except PROGRAMMED INPUT switch bit 1 on, in order that the bit pattern will be shifted slowly enough to easily see what the game program is doing. Once the program has been started, the rate at which the bit pattern is shifted to the left is not affected by any further movement of the front panel switches. From this time on, any time any one of the eight switches in the PROGRAMMED I/O group is changed, then the bit in the PROGRAMMED OUTPUT port which is directly above that switch at the moment is was moved, will change. If it was off before, it will turn on; and if it was on before it will turn off. The direction of travel of the switch is not significant--only that its position was changed. After a switch change is detected, and the light above it turned on or off as

appropriate, no further switch movements will affect the condition of any of the lights until the next shift to the left has occurred. This was done to give the switches time to stop bouncing and stay closed as the processor in this machine is quite fast enough to see the slight bouncing of the switch contact when it initially closes.

By waiting for the next data shift before recognizing any more switch changes, we are prevented from falsely interpreting a bouncing contact as a switch which was repeatedly opened and closed. The object of the game can be either to turn out all the lights in the shifting bit pattern by moving a switch when the bits are passing directly over it, or alternately to turn on all the bits in the shifting bit pattern by moving a switch when a bit which is off is directly over it. Any time the shifting bit pattern is all 0's or all 1's, no movement will be seen in the PROGRAMMED OUTPUT port but by moving any switch, one of the lights will be changed so that the motion is again apparent.

Players can compete for the shortest time to go from all 0's to all 1's, or the other way - from all 1's to all 0's. When the game has been mastered at one rotation speed, the switches can be set for a higher binary number and the system reset to cause the processor to go back to memory location 0 and begin execution of the program again, and a new switch setting will be read to result in a higher rate of rotation, which makes it harder to move a switch at the exact instant the bit desired to be changed is directly above it. If there were only single light on circulating across the output port, and the player, (in attempting to turn it off by moving the switch when the bit was directly over that switch) was too slow, then the bit will have shifted away so that it is now over the next switch to the left, not only will that bit not be turned off, but the bit behind will be turned on so that now there are two bits circulating across the register and the player is further away from achieving all bits turned off.

Knowledge of some of the internal structure of the 8080 processor will be necessary to understand the game program. The Intel data book contains complete information and functional specifications on the internal structure of the 8080 processor, but only the basic aspects of the structure need be known to understand the program operation.

Figure 1 shows the structural blocks in the processor which are important to the programmer. Central to the processor's operation is the register named the ACCUMULATOR. This register and all the others is like one eight bit position in memory or a small "blackboard" with room for only eight bits of either 1's or 0's to be written. When the input instruction was executed during programs 1 and 2, the pattern from the switches on the front panel was read into the ACCUMULATOR register, and when the OUTPUT instruction was given it was again the contents of the ACCUMULATOR which was output to the PROGRAMMED OUTPUT port on the front panel. All arithmetic is done in the ACCUMULATOR and, except for special instructions, (to permit other registers to be read to or from memory) all programmed input/output from either memory or input/out interfaces goes to and from the ACCUMULATOR. The INSTRUCTION register is another "blackboard" with room to store the address where it last read a program byte from memory so that when it finished the execution of that step, it can increment that address by one and use it to determine where to get the next instruction.

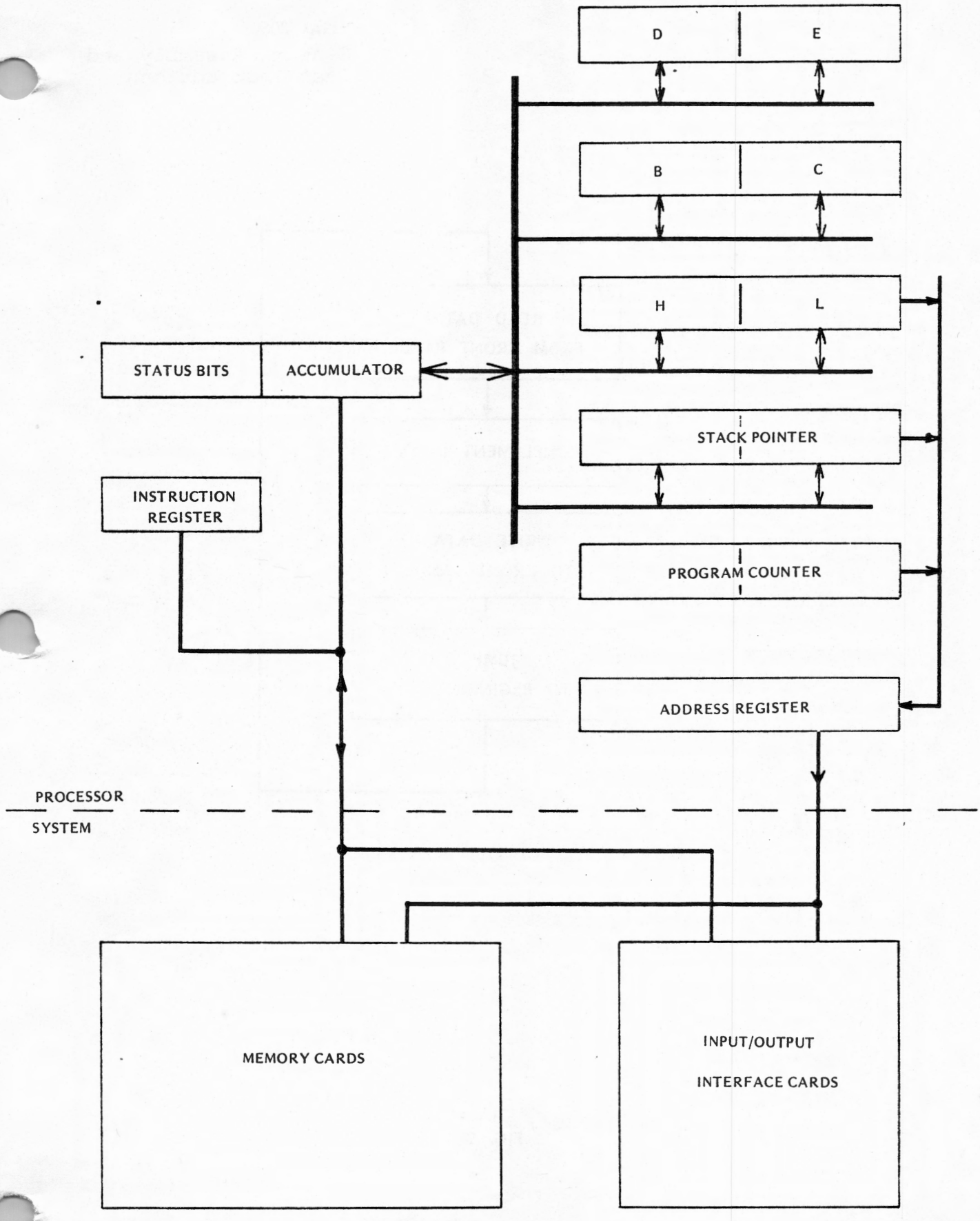
The STATUS BITS are 5 bits that are set to 1 or 0, according to the results of the last data operation performed in the ACCUMULATOR One of the STATUS BITS or condition flags is the Z bit (zero bit) which is turned on when the last operation in the ACCUMULATOR resulted in the ACCUMULATOR being left all 0's. Otherwise, this bit is turned off. The second condition flag is the sign bit. If the most significant bit of the result of the last operation in the ACCUMULATOR has the value 1, this flag is set to 1, otherwise it is reset to 0. Three other condition flags are the sign parity and the auxilary carry, and their functions are described in the Intel Data Book on page 4-2. The fifth condition flag is a carry flag which is turned on if the last arithmetic operation produced an overflow. An overflow is produced, for example, when two numbers are added together and their sum is too large to be contained in the register into which it is put. For instance, if the ACCUMULATOR contained eight 1's and another number was added which contained the value 6, the correct answer would be the combination of the value 5 and a bit turned on in the 9th position. Since the ACCUMULATOR has only eight positions, the carry bit would be turned on.

Some of the STATUS BITS are affected by the operations in other registers than the ACCUMULATOR. For instance the carry bit is affected by additions made in the H and L registers by using the double add instructions. Use is made of this in the game program. There are five other registers in the processor, each of which is 16 bits long, and some of which are divided in half so that operations may be done with only 1/2 at a time. The ADDRESS REGISTER is a 16 bit register over which the programmer has no control. It is simply used to output either the memory address or the input/output address necessary to execute the next cycle. The other four 16 bit registers can all be used by the programmer. There are many instructions in the Intel 8080A processor's instruction set whose function is to move data from any register to any other register, to permit arithmetic operations between a register and the ACCUMULATOR (with the result always being left in the ACCUMULATOR), and some special instructions to permit direct transfer of data from memory to a register, or vice versa.

The B, C, D, and E half registers are all general purpose registers. The H and L register pair and the STACK POINTER register pair both have special functions in addition to being usable for general purposes. The game program does not make use of these special functions.

With the basic structure of the processor in mind, we can now look at the operation of the game program. Larger programs can not be readily understood or written by working directly on the list of machine instructions, such as we did for Test Programs 1 and 2. A flow diagram is essential to quickly follow the sequence of the instructions and understand how they work together to achieve the desired result.

Figure 2 shows a flow diagram for Program 2. Each program function is briefly described in a separate box, and the flow of the execution of the program is indicated by the lines. Test Program 2 was a simple loop with no decisions so that after executing the short sequence of instructions, the flow of the program is back to the beginning of the loop to begin again. Figure 3 shows the flow diagram for the game program. Although it need not be understood to execute the game program, a thorough understanding of how this flow diagram achieves the operation of the game will be a useful step towards writing your own programs.



8080 PROCESSOR
FIG. 1

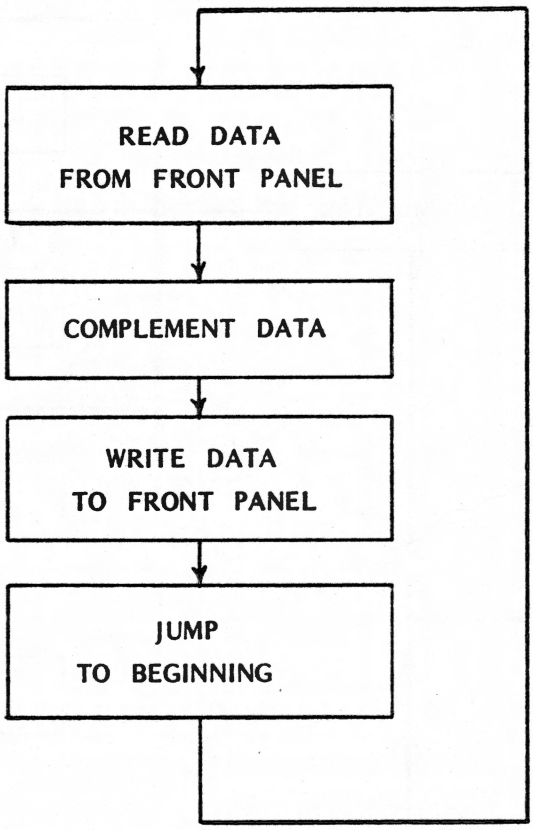

READ DATA
FROM FRONT PANEL
COMPLEMENT DATA
WRITE DATA
TO FRONT PANEL
JUMP
TO BEGINNING

FIG. 2

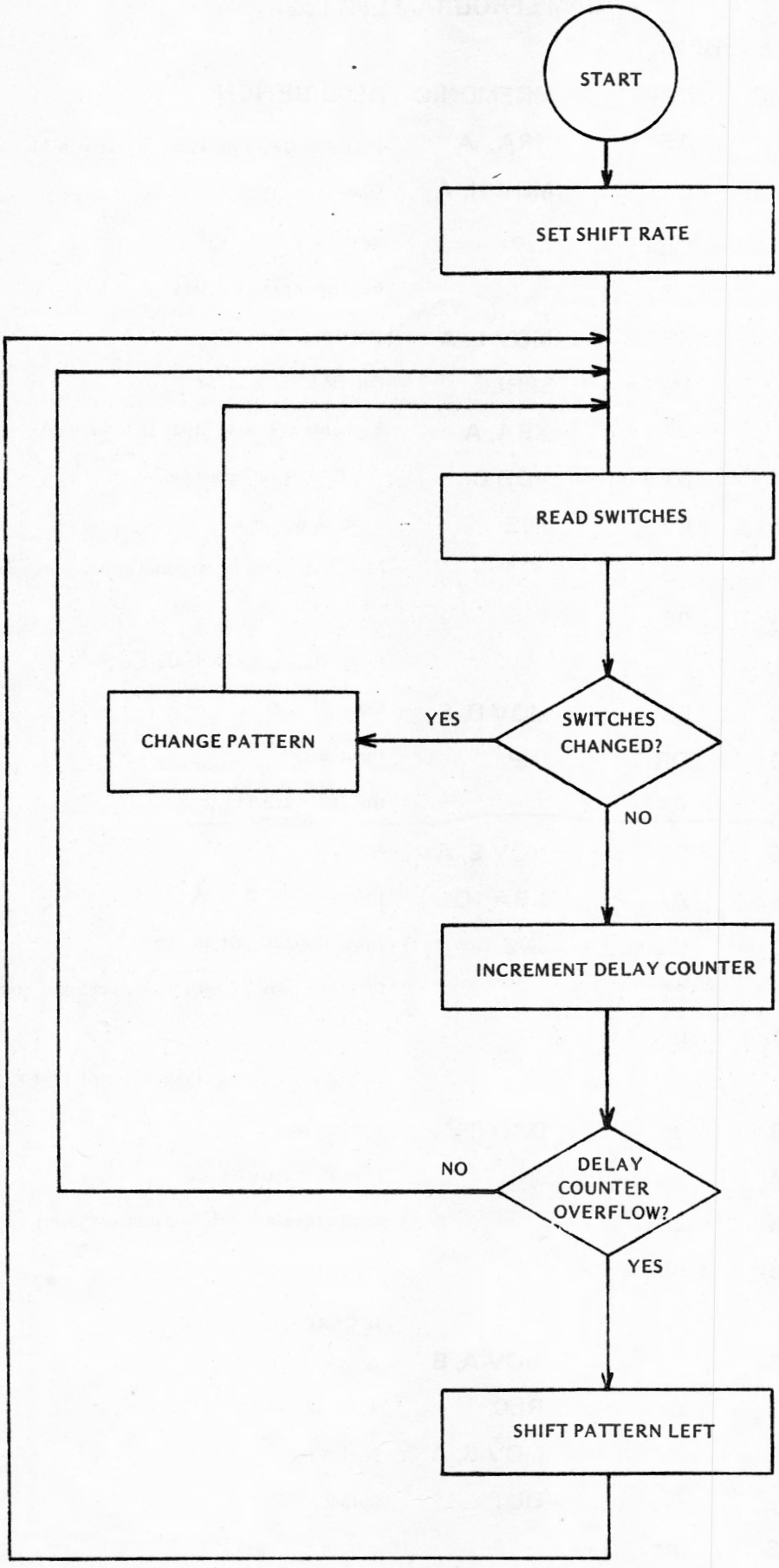


FIG. 3

GAME PROGRAM BASIC FLOWCHART

c 1975 IMS Assoc Inc

GAME PROGRAM LISTING

OCTAL ADD.	OCTAL INST.	HEX ADD.	HEX INST.	MNEMONIC	DESCRIPTION
000 000	257	0000	AF	XRA, A	Exclusive OR A to itself (put zero in A)
001	147	01	67	MOV H, A	Move A to H (put zero in H)
002	333	02	DB	INP	Input data
003	377	03	FF		from front panel switches
004	157	04	6F	MOV L, A	Move A to L
005	371	05	F9	SPHL	Put H&L reg. into SP
006	257	06	AF	XRA, A	Exclusive OR A to itself (put zero in A)
007	201	07	81	ADD C	Put C in A, affecting flag bits
010	302	08	C2	JNZ	Jump if not zero
011	023	09	13		(skip switch test for debounce after a switch change)
012	000	0A	00		
					IF NORMAL, CONTINUE
013	123	0B	53	MOV D, E	Move E to D
014	333	0C	DB	INP	Input data
015	377	0D	FF		from front panel switches
016	137	0E	5F	MOV E, A	Move A to E
017	252	0F	AA	XRA, D	Exclusive OR D to A
020	302	10	C2	JNZ	Jump if result not all O's
021	041	11	21		(change display if switch position changed from last time)
022	000	12	00		
					IF SWITCHES UNCHANGED, CONTINUE
023	071	13	39	DAD SP	Add SP to HL
024	322	14	D2	JNC	Jump if no carry results
025	006	15	06		(return to read switch loop if no carry yet)
026	000	16	00		
					IF CARRY, CONTINUE
027	170	17	78	MOV A, B	Move B to A
or 030	007	18	07	RLC	Rotate left 1
030	017	18	0F	RRC	Rotate Right 1
031	107	19	47	MOV B, A	Store A in B
032	323	1A	D3	OUT	Output A
033	377	1B	FF		in front panel lights
034	257	1C	AF	XRA, A	Exclusive OR A to itself (put zero in A)
035	117	1D	4F	MOV C, A	Move A to C (Reset debounce indicator)

© 1975 IMS Assoc INC

GAME PROGRAM LISTING (CONT.)

OCTAL ADD.	OCTAL INST.	HEX ADD.	HEX INST.	MNEMONIC	DESCRIPTION
036	303	1E	C3	JMP	Jump
037	006	1F	06		(to read loop)
040	000	20	00		
					CHANGE DISPLAY IF SWITCH DIFFERENT
041	250	21	A8	XRA, B	Exclusive OR B with A
042	107	22	47	MOV B, A	Store A in B
043	323	23	D3	OUT	Output A
044	377	24	FF		in front panel lights
045	257	25	AF	XRA, A	Exclusive OR a with itself A (put zero in A)
046	147	26	67	MOV H, A	Move A to H (set counter to insure enough delay for debounce)
047	057	27	2F	CMA	Complement A (to all I's)
050	117	28	4F	MOV C, A	Move A to C (set C to debounce)
051	303	29	C3	JMP	Jump
052	006	2A	06		(to read loop)
053	000	2B	00		

NOTE:

Exclusive OR of two switch patterns results in I's in positions which were changed, with all 0's elsewhere.

B= DISPLAY BYTE STORAGE

C=SWITCH DEBOUNCE INDICATOR
I=DEBOUNCE 0=NORMAL OPERATION

D=LAST SWITCH SETTINGS

E=CURRENT SWITCH SETTINGS

H,L=DELAY COUNTER

SP=INCREMENT FOR DELAY COUNTER

c) 1975 IMS Assoc Inc

Cabinet Assembly

CABINET ASSEMBLY INSTRUCTIONS

Begin by installing the correct number of plastic card guides on the chassis part C's. The card guides should be placed from the front backwards, an equal number on each piece C, taking care that the wedge - shaped opening of the slot is positioned upwards. Note that the two ends of piece C are not symmetrical. The end with the wider space between the last small hole for mounting the card guide and the end flange is placed toward the back of the cabinet, so that the guides will line up with the connectors on the Mother board.

The card guides should be assembled starting from the front end (with card guide mounting holes placed closer to the end flange). Make sure you place the card guide so as to form a left hand and a right hand mounting-rail piece. If this is not done, then the card guides will be upside down on one of the two piece 'C's when they are mounted into the cabinet. The card guides are most easily mounted using a small press and placing the tab of each card guide in position started into the hole and pressing them into place until the mounting tabs snap through. A drill press with a large flat - headed screw mounted in the chuck works well with this operation. (Alternately, the card guides may be installed very carefully using needle nosed pliers). Care should be taken that the tabs are started into the hole when beginning to press the guide into place, otherwise one or both may be bent out flat and broken off. One end of the guide at a time should be inserted rather than trying to press both ends in simultaneously.

If a fan is to be installed in the chassis, it should be assembled on the back frame piece A1 at this time using the hardware in the fan kit. The fan terminals should be towards the top and towards the Mother board side of the chassis.

Next, the power cord should be inserted using the special grommet in the hole provided on the back panel. 4 to 6 inches of the power cord should be left on the inside of the cabinet. If the power cord grommet is squeezed together with a pair of pliers before insertion into the cabinet back, it will ease the job of inserting this tight fitting grommet. To insert the grommet, the power cord should be pulled through the hole nearly to the point where the grommet has been placed around the power cord, then the outer edge of the grommet can be grasped with a pair of pliers and squeezed slightly and inserted in the hole and worked in while slight tension is also being put on the cord from the back side to assist. Working this grommet in by rocking it back and forth works better than just pushing harder.

The front and back frames can now be screwed to the base plate using 6-32x5/16" machine screws. Note that the back frame fits <u>under</u> the base plate and the front frame fits on <u>top</u> of the base plate, set back about 1" from the front edge of the base plate. Next, install

the two card frames between the front and back frames. Use two 6-32x5/16" machine screws at each end of each card frame. The front and back frames have slotted holes allowing the card frames to be adjusted slightly when the Mother board is installed on the base plate and boards are inserted in the card frames.

The self-adhesive rubber feet can then be separated from each other, the protective backing removed, and placed on the bottom of the cabinet spaced 3 inches along the left hand and right hand edge of the bottom, to support the cabinet weight.

BASE PLATE HOLE IDENTIFICATION

The base plate currently being shipped is a universal base plate, with extra holes for accomodating two styles of mother board mounting systems and two styles of power supplies. For the power supply and mother board systems shipped with your kit:

1. Place the Power Supply p.c. board in the base plate cavity and line up the holes in the p.c. board with the corresponding holes in the base plate and mark (e.g., with a felt-tip pen) which holes are to be used.

2. The mother board mounting system uses the two rows of 12 holes each on the left side of the base plate.

SWITCH ESCUTCHEON INSTALLATION

When the CP-A Front Panel Assembly has been mounted, the Switch Escutcheon (piece A2B) can be installed on the base plate at the front of the computer using four 6-32x5/16" Phillips pan head machine screws. Note that the Escutcheon should fit under the base plate.

CABLE CLAMP INSTALLATION

Cables that do not fit the connector holes on the back frame of the chassis may be clamped for strain relief at the top of the back frame using the L - shaped aluminum bar, piece K. Install using two 6-32x½" Phillips pan head machine screws. Depending on the thickness of the cables being clamped, either of the two sides of the angle may be used.

TABLE TOP COVER INSTALLATION

To install the table top cover, slide the cover carefully over the chassis frame and hold in place with four 6-32x½" Phillips pan head machine screws.

Refer to Appendices for an exploded view of the chassis cabinet.

RACK MOUNT SYSTEM ASSEMBLY INSTRUCTIONS

For the rack mount system, begin by installing the rack mount cover on the chassis. Use five 6-32x5/16" Phillips pan head machine screws. Next install the left and right side plates to the chassis with the front flanges pointing outwards. Use four 6-32x½" Phillips pan head machine screws on each side plate. The forward holes in each pattern on the side plates should be used.

Next mount the assembly in the rack using two screws on each side of the front flanges. Hardware requirements for mounting the assembly into the rack will vary according to the individual rack. It is suggested that the rear of the assembly also be supported in the rack. Finally, mount the front face panel onto the side plate flanges using four #10 round head screws and clips.

NOTE: for installations without slides where easy removal of the computer is desired, the side plates can be mounted directly in the rack and the computer can be slid on its rubber feet on the bottom flanges of the side plates. The rear of the side plates in this case should be fastened securely to the back of the rack cabinet.

UNIT: CABINET

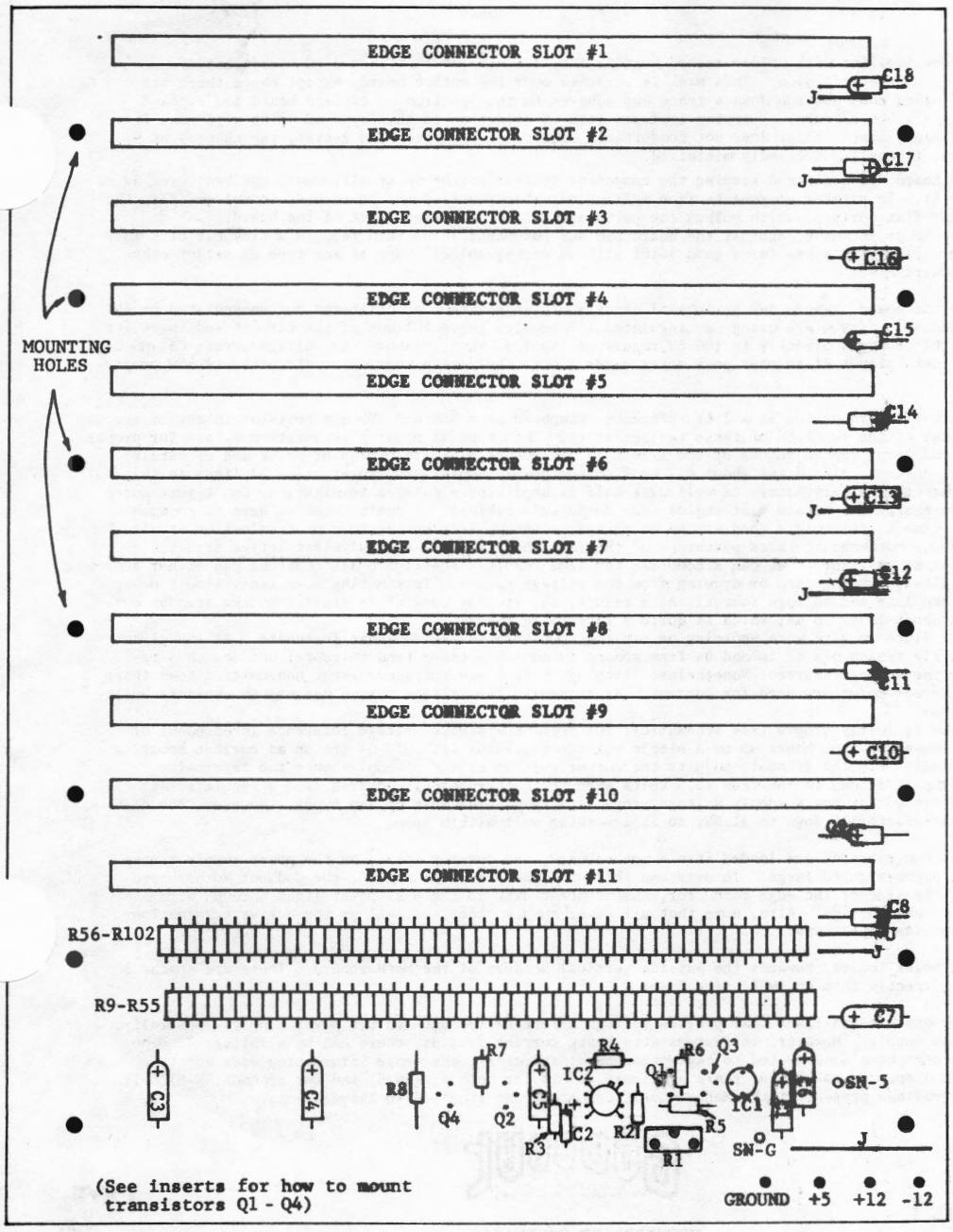
ITEM	IMSAI PART #	QUANTITY	DESCRIPTION/IDENTIFYING MARKS
A1		1	Backframe
A2B		1	Painted Frontpiece
(A2A)		(1)	(Painted Frontpiece - for OEM Unit only
A3		1	Cabinet Base
B		1	Front Frame
C		2	Card Frame
K		1	Cable Clamp
S		1	Capacitor Brace (with foam tape)
h-s-6-32x5/16"	20-3202001	21	6-32x5/16" Phillips pan-head machine screws
h-s-6-32x½"	20-3502001	2	6-32x½" Phillips pan-head machine screws
rubber feet	28-040001	6	Self-adhesive rubber feet, gray

UNIT: TABLE TOP COVER

ITEM	IMSAI PART #	QUANTITY	DESCRIPTION/IDENTIFYING MARKS
J		1	Table Top Cabinet Top
h-s-6-32x½"	20-3502001	4	6-32x½" Phillips pan-head machine screws

ITEM	IMSAI PART #	QUANTITY	DESCRIPTION/IDENTIFYING MARKS
Back Frame	93-3010003	1	Back Frame "A1"
Front Piece	93-3010010	1	Painted Front Piece "A2B"
Cabinet Base	93-4010004	1	Cabinet Base "A3"
Front Frame	93-3010001	1	Front Frame "B"
Card Frame	93-3010002	2	Card Frames "C"
Cable Clamp	93-3010013	1	Cable Clamp "K"
"DANGER" Label		1	"DANGER" Label on right card frame
Rubber Feet	28-0400001	6	Adhesive backed rubber feet
Screw	20-3302001	25	6-32x5/16" Phillips pan head machine screws
Screw	20-3502001	2	6-32x½" Phillips pan head machine screws
Capacitor Brace	91-0300001	2	Capacitor Braces with foam backing

Mother Board



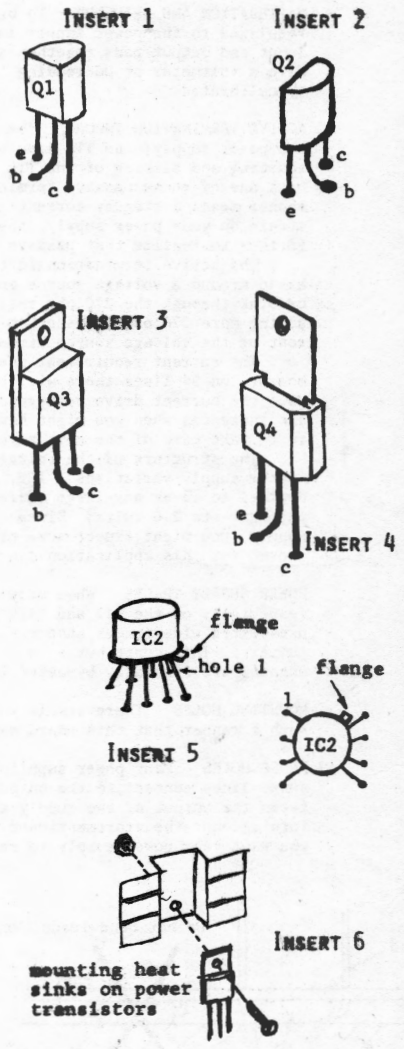
COMPONENT LAYOUT OF MOTHERBOARD

(TOP VIEW)

Kit No. 105 Motherboard

GODBOUT

BILL GODBOUT ELECTRONICS
BOX 2355, OAKLAND AIRPORT, CA 94614



ORDER OF ASSEMBLY...Read these instructions carefully, take your time and don't rush; after about 3½ hours you will end up with a quality motherboard. (1) Check parts against the parts list. (2) This board uses a solder mask that virtually makes solder bridges a thing of the past. However, solder masked boards require a specific style of soldering, so flip this sheet over and read the section on soldering techniques before proceeding. (3) Mount resistors R56 - R102; flip the board over as shown under soldering techniques, then solder all leads and clip off excess lead length. (4) Mount resistors R9 - R55, flip the board over, then solder all leads and clip off excess lead length as in the previous step. (5) Mount and solder all other resistors except the 10K trimpot in a manner similar to the previous 2 steps. (6) There are 7 jumpers on this board; locate all 7, then install and solder jumper wires Use #22 gauage or larger wire. (7) Install the capacitors on the board, carefully observing polarity. We recommend mounting the capacitors so that the values face upward and are easily readable. (8) Mount and solder the 10 edge connectors. Note that there are 11 slots; you may install the edge connectors in any 10 of the slots. When using this board as an add-on to machines like the IMSAI, one of the slot positions may be used for a jumper between the existing motherboard and the Godbout motherboard. (9) Straighten the leads of the 10K trimpot, then solder in place. (10) Install IC1, the three terminal regulator Solder carefully and quickly in this and the next step to prevent possible heat damage to the ICs (11) Spread the leads of IC2 in accordance with Insert 5; note that the lead associated with the flange would be pin 8 of a mini DIP package. When viewed from the top, the lead to the immediate left of the flange inserts into the hole marked "1" on the circuit board. (12) Install and solder transistors Q1 and Q2, orienting the leads as shown in inserts 1 & 2 (13) Mount the heat sinks on the two power transistors in accordance with Insert 6, using 2 of the 6 nut/bolt combinations provided with your kit. Then, install and solder these transistors as shown in Inserts 3 & 4. PLEASE NOTE. Transistors Q1 - Q4 must be properly oriented for your kit to work properly. Solder quickly and carefully as semiconductors are heat sensitive. (14) Attach the 4 solder lugs provided, using the remaining hardware, to the 4 power line connections located near the trimpot. (15) Finally, examine your work carefully (magnifying glasses are a big help in checking over connections). When you are satisfied that all is well, turn to the other side of this sheet for calibration procedure and available options.

PARTS LIST

R1	10K trimpot
R2	150K resistor (brwn/grn/yel)
R3,5	560 ohm resistor (grn/blu/brwn)
R4,8	1K resistor (brwn/blk/yel)
R6,7	910 ohm resistor (wh/brwn/brwn)
R9-R102	270 ohm resistor (red/violet/brwn)
C1,3,4,5, 6,7,10, 13,16	39 uF tantalum capacitor
C2	0.47 uF tantalum capacitor
C8,9,11, 12,14,15, 17,18	2.7 uF tantalum capacitor
Q1	PNP transistor F137 (2N2907A)
Q2	NPN transistor 3560-2/PT134 (2N3904)
Q3	NPN power transistor (GE red)
Q4	PNP power transistor D41D1
IC1	Three terminal regulator (G309H)
IC2	Micropower op amp (G4250H)
(2)	THM6070 heat sinks
(1)	Circuit board
(10)	S-100 edge connectors
(4)	Solder lugs
(6)	Nuts & bolts
(1)	Data sheet

SOLDERING TECHNIQUES In case you are not familiar with solder masked boards, they are similar to standard PC boards but are screened with a solder resistant coating on the foil side. This mask is screened over the entire board, except where there are solder connections to be made. Thus, solder does not run down a trace but adheres to the junction of exposed board and exposed lead. Because of this, soldering requires a bit of care; trying to force lots of solder on to this type of joint will make it ball up around the lead. On the other hand, since solder does not comfortably hold to or flow over the resist, the chances of getting a bridge between tight, adjacent traces are decidedly minimized.

When soldering with a solder mask board, we recommend keeping the component leads straight up at all times, not bent over as with other types of boards (see figure 1). To prevent components from falling out when you flip the board over to solder, flip the board over on to a table, book, or other flat surface, which pushes the parts against the component side of the board.

When soldering, bring the iron tip in at an angle, against the board pad and component lead; then feed in a tiny bit of solder at opposite ends of the lead (see figure 2). This makes for a good joint with no excess solder. Use of any type of solder other than rosin core solder invalidates the warranty.

CALIBRATION AND BRING-UP To bring up the board, remove any peripheral cards and attach either +8V filtered but unregulated or +5V regulated to the power supply lug marked +5. If you are using a regulated +5V supply, leave IC1 out of the circuit and short its input and output pads together so that R1 connects directly to the 5V regulated supply. Then, measure the voltage across C3 or C4 with a voltmeter or DC reading 'scope; set trimpot R1 so that your meter reads 2.6V. The active terminator circuitry is now properly calibrated.

ACTIVE TERMINATION THEORY The standard TTL termination is a 2.6V reference, composed of a 360 and 390 ohm resistor in series across the power supply; the TTL line terminates at the junction of these two resistors. This type of passive termination allows for proper sourcing and sinking of the TTL line, and keeps the impedance of the line to a minimum to minimize pickup of noise and crosstalk. Each one of these passive terminations, however, also draws about 6.7 mA from the power supply. So, terminating 94 lines in this manner means a standby current drain through the terminators of well over half an amp! These passive terminations don't just put a strain on your power supply, they waste energy and create heat inside your computer's cabinet. I don't think we have to go much further to realize that passive termination is not such a good way to do things, although it's better than no termination at all.

The active termination in the Godbout Motherboard takes advantage of the fact that there is an equivalent active structure, based around a voltage source and isolating resistor, that can accomplish the same results (see figure 3). Current can either source or sink through the 270 ohm resistor, either dumping into or drawing from the voltage source. Terminating more lines simply means adding more 270 ohm resistors between the line and voltage source. As a result, the standby current is slashed to the standby current of the voltage source circuitry---about 15 or 20 mA, which is quite a savings in energy.

The current requirement goes up as lines require more sourcing or sinking, but here we are somewhat fortunate. At any given moment, on 94 lines there will be a fairly random mix of 1s and 0s from moment to moment---these tend to cancel out and thus reduce the current drive requirements of the voltage source. Nonetheless, although this keeps average current consumption down there are instances when you might have an extreme momentary need for current. As a result, the voltage source has enough capacity built-in to take care of the most adverse cases.

The structure of the voltage source is fairly simple (see schematic); IC1 sets up a stable voltage reference independent of master supply variations. IC2, a micropower op amp, hooks up as a simple voltage regulator with Q1 - Q4 set up as current boosting devices to cover any large current demands. R1, the trimpot, adjusts the output voltage of the op amp---hence the terminator voltage---to 2.6 Volts. Since the op amp is forced to run from ±2.5 Volts when the motherboard is powered from a regulated +5V supply, you might expect some problems since that low a supply voltage range is marginal for most op amp types. However, the 4250 chosen for this application can work satisfactorily down to ±1.5V, so it's working well within spec.

POWER SUPPLY TRACES When many current-hungry cards are loaded into a motherboard, the current strain on the power supply traces (especially on the +12 and -12V lines) becomes quite large. To overcome this unbalanced current demand, the Godbout Motherboard uses extra wide traces that run off to the side of the edge connector, then connect over to the buss power lines through wire jumpers. This contributes to a more reliable design. Also, note that all power supply lines, as well as the active termination output, are liberally bypassed with tantalum capacitors.

MOUNTING HOLES There are 14 mounting holes located towards the outside vertical borders of the Motherboard. These are spaced in such a manner that this board may bolt directly into an IMSAI size frame.

SENSE HOLES Many power supplies incorporate sense lines that provide voltage reference feedback to the power supply. Generally, sense lines connect to the output of the supply. However, in systems with heavy current demands, there can be a voltage drop between the output of the supply and the peripherals connected to the Motherboard; therefore, the sense information does not take into account the aforementioned voltage drop. On the board, there are sense points for both +5 (SN-5) and for ground (SN-G). If you want your power supply to read the voltage present at the Motherboard, connect sense lines up to these points.

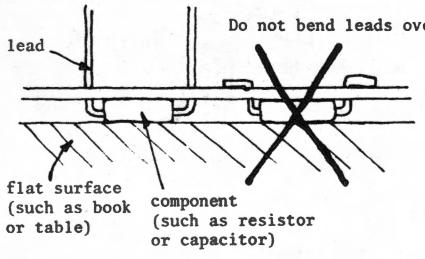


FIGURE 1

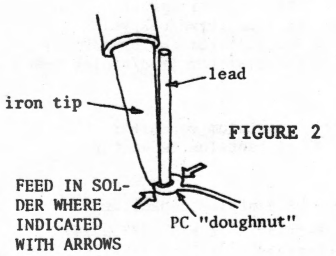


FIGURE 2

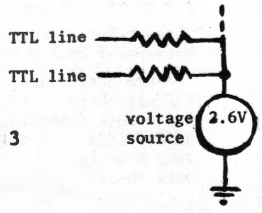
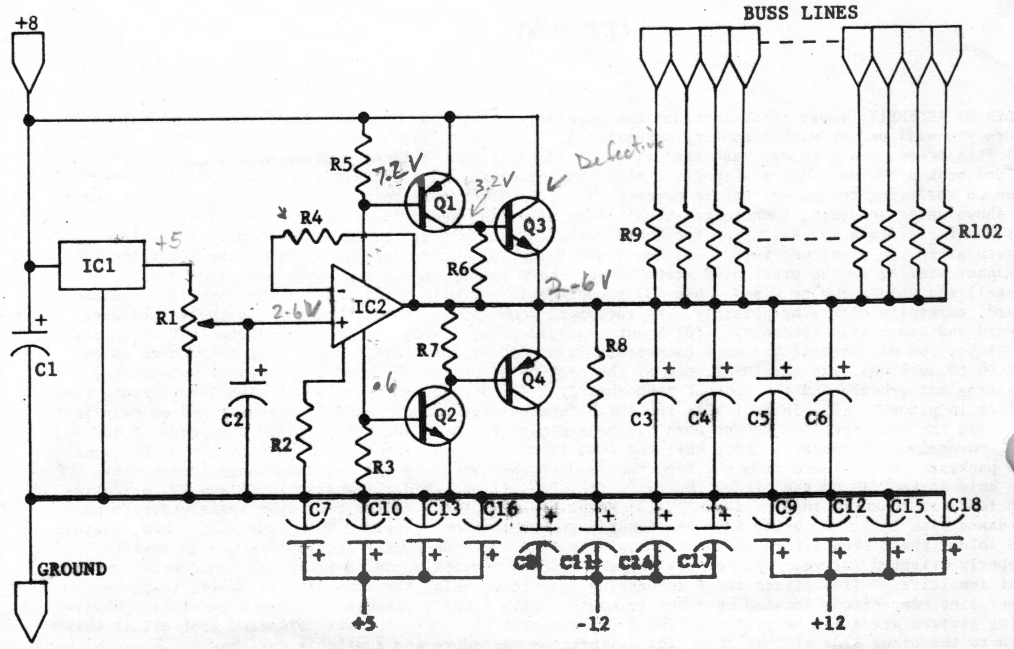


FIGURE 3

GODBOUT
BILL GODBOUT ELECTRONICS
BOX 2353, OAKLAND AIRPORT, CA 94614

Op amp ±3.4V

MOTHERBOARD SCHEMATIC



MOTHER BOARD

FUNCTIONAL DESCRIPTION

The IMSAI 8080 system Mother boards are available in three different length sections varying from a minimum of 4 printed circuit card connector positions. The basic system includes a Mother board with six connector positions on it. One is used for the front panel and the other five are available for the MPU and any combination of memory or I/O cards.

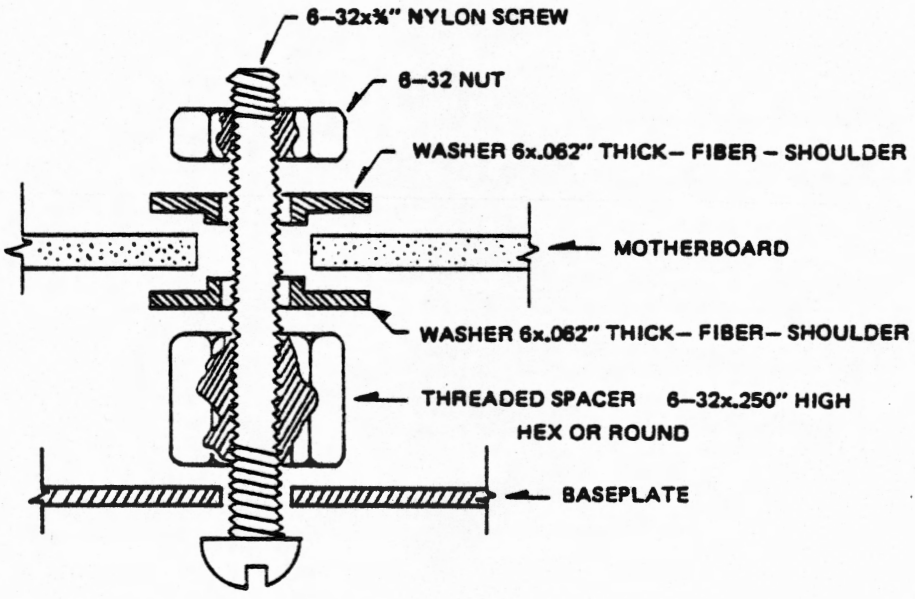
The card-to-card spacing on the Mother board is 3/4 inch except for the front position which is reserved for the front panel board or the parallel I/O board for the dedicated processor to accommodate mounting the card in the special front position in the cabinet.

Additional sections of Mother board are available with positions for 4 connectors. These may be added to the system at a later date, and connected to the previous Mother board sections by jumpers between the sections soldered into provided holes. No jumper wire soldering is required if the full-length board is purchased.

The Mother board is 1/16 inch printed circuit board with double-sided plated-through holes. Each of the connector pins is connected by traces on both sides of the board. Heavy power traces are provided to handle the very large currents involved in a fully-loaded back plane. The two connectors supplied with the IMSAI system are high-quality gold-plated-contact connectors, for reliable contacts and long life.

Trace spacing is tightly controlled on the board to avoid any close spots where shorts from solder bridges might tend to occur. The traces on Mother board are plated for better appearance and more reliable solder connections. A solder mask is provided on both sides of the Mother board.

MOTHERBOARD MOUNTING SYSTEM



4 PER EXP–4

8 PER EXP–8

24 PER EXP–22

MAINFRAME ASSEMBLY
FIGURE 2

MOTHER BOARD

ASSEMBLY INSTRUCTIONS

The Mother board appears to be the simplest of all the boards to assemble. The solder mask minimizes the chances of shorting adjacent traces. However, it is imperative that extra care be taken during assembly to avoid excess solder shorting adjacent pins. Because a short on the Mother board is extremely hard to locate and correct when it is between the board and the connector, it is worth the builder's time to give special attention to making certain that no such shorts occur. Use only as much solder as required for a good joint. If too much solder is used, either the pool of solder can short to an adjacent pin on the top side or the solder can leak through and form a ball on the backside which can also short to an adjacent pin.

The board should be checked with an ohmmeter carefully both before and after assembly to insure that it will operate properly. While the chance that incomplete etching during manufacture left two traces shorted is extremely slight, the ohmmeter check before assembly is worth while simply because it would be so difficult to correct such a problem after a socket is soldered in place over it.

To test the board, either a simple ohmmeter or a battery connected to a buzzer or a light bulb and test leads are all that is required. Each pair of adjacent traces should be checked with the continuity tester to be sure that there is no connection between them. Should any adjacent traces be found to be electrically connected during this pre-assembly check, careful inspection of the board should reveal the short. Any imcompletely-etched copper or other metallic path between the two traces shold be removed with a sharp knife, such as an X-acto knife.

After each connector is soldered in, the continuity check should be made again to make certain that during assembly no shorts were created. If any are discovered, steps should be taken to remove them before further assembly. In most cases, this short will have been caused by too much solder having been applied and may be removed simply by removing the excess solder. If an Extender board is available, a simple tester may be made from it by temporarily connecting all the pins on the front side, except pin 1, together, connecting all the pins on the back side, except pin 100, together and then connecting the continuity tester between the two sides of the Extender board. If this extender board is inserted in the socket as it is being soldered, the continuity tester will indicate immediately any short between any two adjacent traces.

Mother Board
Parts List

EXP-4

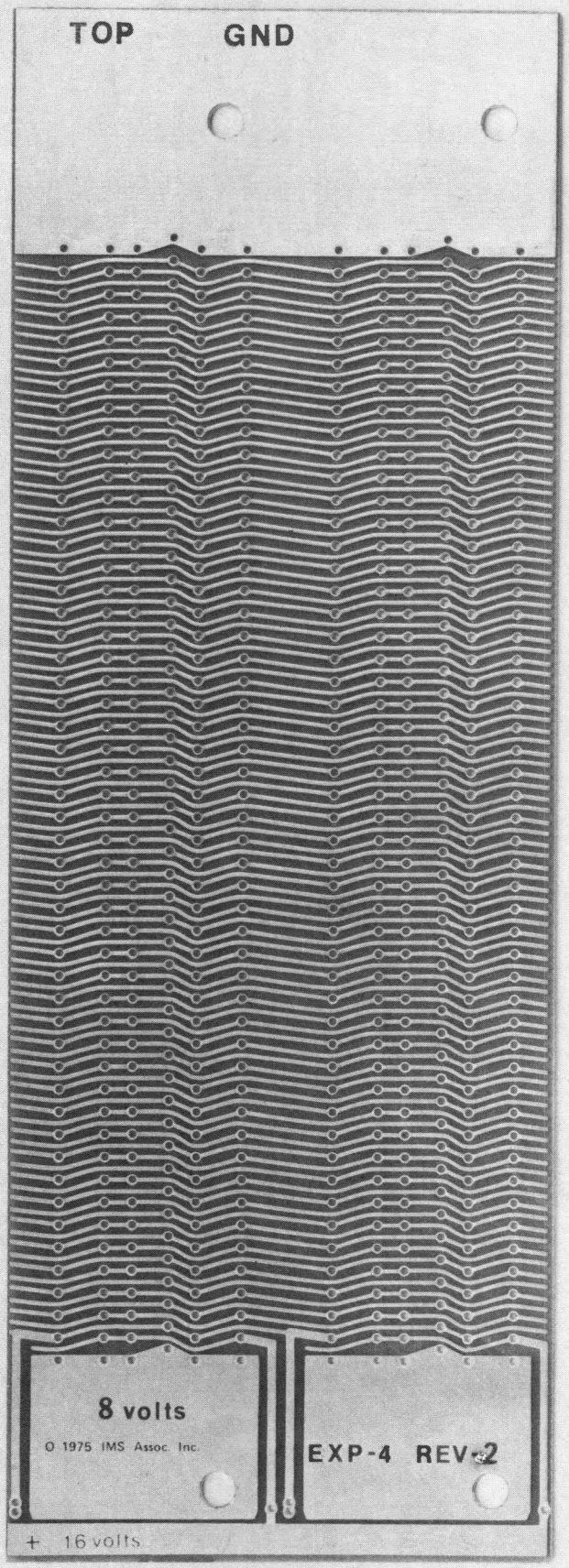
ITEM	IMSAI PART #	QUANTITY	DESCRIPTION/IDENTIFYING MARKS
PC Board	92-0000004	1	4-Slot Printed Circuit Board
Washer	21-3330001	8	#6 Sholder Fiber Washer
Spacer	21-4600001	4	6-32x¼" Threaded Spacer
Nut	21-3120001	4	6-32 Nut
Screw	20-3701002	4	6-32x3/4" Nylon Screw

EXP-6

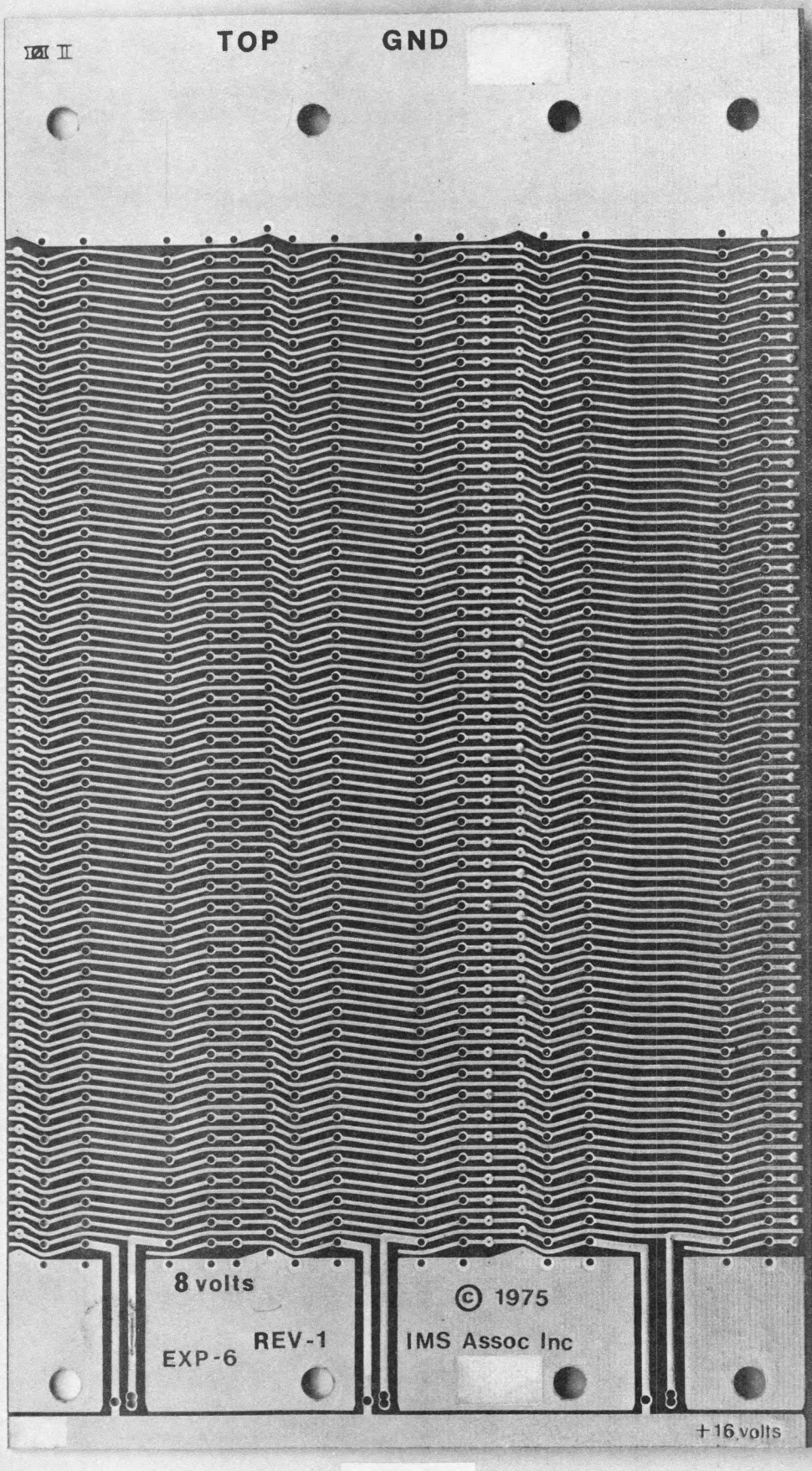
PC Board	92-0000005	1	6-Slot Printed Circuit Board
Washer	21-3330001	16	#6 Shoulder Fiber Washer
Spacer	21-4600001	8	6-32x¼" Threaded Spacer
Nut	21-3120001	8	6-32 Nut
Screw	20-3701002	8	6-32x3/4" Nylon Screw

EXP-22

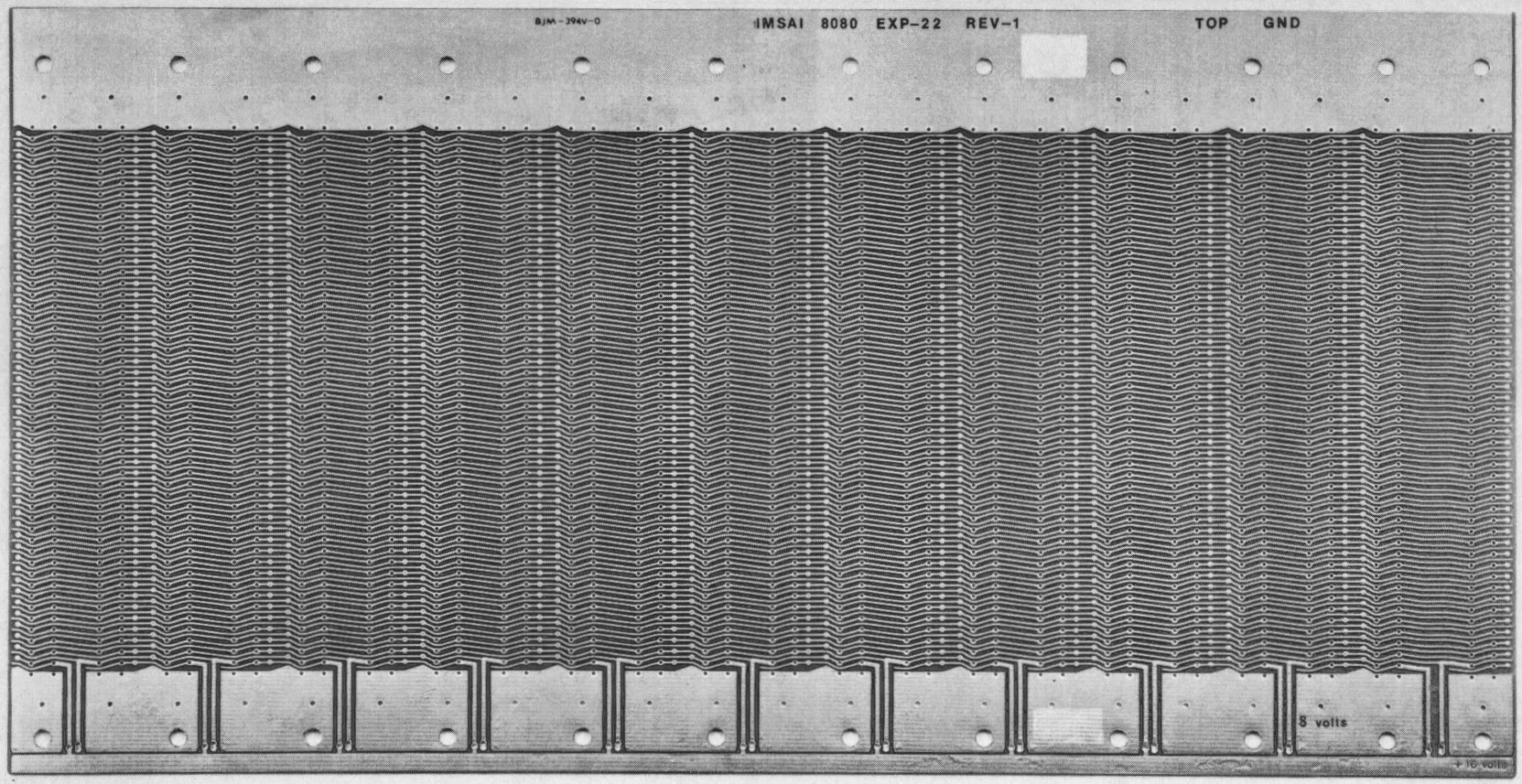
PC Board	92-0000006	1	22-Slot Printed Circuit Board
Wahser	21-3330001	48	#6 Shoulder Fiber Washer
Spacer	21-4600001	24	6-32x¼" Threaded spacer
Nut	21-3120001	24	6-32 Nut
Screw	20-3701002	24	6-32x3/4" Nylon screw



EXP-4 Rev. 4



EXP-6



EXP-22 Rev. 1

SOCKET INSERTION

The 100 pin edge connectors are symmetrical so that they may be inserted either way. The connector stands off the board slightly supported by raised feed at each end. Each connector should be checked during assembly to make sure that it is seated properly and that the Mother board near the center of the connector is neither pushed further toward the connector nor lifted away before the connector is soldered in place to prevent the Mother board from bowing.

The Mother board is not completely symmetrical and the connectors must be inserted from the top side. The top side is the side on which the +8 volt foil is broken every 2 connectors to allow the 2 traces for + and -16 volts to extend from the 16 volt bus at the end of the board into the connector pins. The back side of the board has both the very heavy ground bus and the 1 inch wide 8 volt foil area continuous for the full length of the board. The +16 volt trace is the .2 inch trace on the edge of the board alongside the +8 volt bus on the front side, that is, the side where the +8 is broken to allow for the pairs of +16 volt traces to extend into the pins. The -16 volt bus is the .2 inch wide trace along the edge of the board on the back side underneath the +16 volt bus. NOTE: Before mounting any connectors, locate the front of the Mother board. The connector for the front panel (CPA board) needs to be mounted in the first position at the front of the Mother board. Notice that the spacing between the first and second positions at the front of the Mother board is wider than the spacing between any two of the other connector positions.

The suggested procedure for inserting and soldering a connector is to insert the connector in place, seat the two ends firmly against the feet and solder the two pins on each end.

Next, the position of the center of the Mother board next to the connector should be checked and either pushed further toward the connector or pulled away so that the gap between the connector and the Mother board is uniform all the way across. Then the two pins in the center of the connector should be soldered.

One final check should be made to make sure that the gap is uniform all the way across the connector and the remaining pins in the connector should be soldered.

Care should be taken to check each connector after solder to make sure that every pin was soldered because it is easy to miss a pin and not see it during a quick inspection. After the last connector is soldered in place and the board checked you are ready to install the power connections and mount the board in the cabinet.

See MAINFRAME ASSEMBLY section for connecting the Mother board to the Power Supply and mounting the Mother board in the chassis.

Mother Board

USER GUIDE

With the proper care taken during assembly, the Mother board should be the most reliable board in the system. The only attention the user will typically put on the Mother board, is when he desires to add more card slot positions. Either 4 slot extension Mother Boards may be added to the original 6 slot Mother Board, or the 6 slot board may be replaced by a new 22 slot board.

If 4 slot extension(s) are used, the extension(s) should be assembled according to instructions for assembling the original Mother board. Then the original Mother board must be removed from the cabinet and jumpered to the new section by the use of short wire jumpers between the connection points provided in each trace.

The power buses should be connected with a much heavier wire. The two boards can then be reassembled into the cabinet.

Care should be taken when inserting jumpers that each goes between the corresponding lines on the two sections of Mother board.

IMSAI 8080 BUS SIGNAL LIST

1	+8v
2	+16v
3	XRDY
4	VI Ø
5	VI 1
6	VI 2
7	VI 3
8	VI 4
9	VI 5
10	VI 6
11	VI 7
12	
13	
14	
15	
16	
17	
18	STATUS DSBL
19	CCDSBL
20	**
21	SS
22	ADDR DSBL
23	DO DSBL
24	Ø2
25	Ø1
26	PHLDA
27	PWAIT
28	PINTE
29	A 5
30	A 4
31	A 3
32	A 15
33	A 12
34	A 9
35	DO 1
36	DO Ø
37	A 10
38	DO 4
39	DO 5
40	DO 6
41	DI 2
42	DI 3
43	DI 7
44	SMI
45	SOUT
46	SINP
47	SMEMR
48	SHLTA
49	CLOCK (2 MHz)
50	GND

51	+8v
52	-16v
53	SSW DSB
54	EXT CLR
55	*
56	
57	
58	
59	
60	
61	
62	
63	
64	
65	
66	
67	
68	MWRITE
69	****
70	***
71	RUN
72	PRDY
73	PINT
74	PHOLD
75	PRESET
76	PSYNC
77	PWR
78	PDBIN
79	A Ø
80	A 1
81	A 2
82	A 6
83	A 7
84	A 8
85	A 13
86	A 14
87	A 11
88	DO 2
89	DO 3
90	DO 7
91	DI 4
92	DI 5
93	DI 6
94	DI 1
95	DI Ø
96	SINTA
97	SWO
98	SSTACK
99	POC
100	GND

* reserved for chassis ground
** reserved for memory unprotect
*** reserved for memory protect
**** reserved for protect status

IMSAI 8080 SYSTEM BUS STRUCTURE

The IMSAI 8080 system bus structure consists of 100 lines. These are arranged 50 on each side of the plug-in boards, with pins 1 through 50 on the component side and pins 51 through 100 on the back side. As the board is viewed right-side up (components up, 100 pin connector towards you) pin #1 is on the left end on the top and pin 51 is on the back side directly opposite pin #1.

Conventions:

SYMBOLS: "P" prefix indicates a processor command or control signal

"S" prefix indicates a processor status signal

LOADING: All inputs to a card should be loaded with a maximum of 1 TTL low power load

LEVELS: All bus signals except the power supply are TTL. All Data and Address lines are positive TRUE (ground = 0 bit)

BUS DEFINITION

Front Side

No.	SYMBOL	NAME	FUNCTION
1	+8V	+8 volts	Unregulated input to 5v regulators
2	+16V	+16 volts	Positive unregulated voltage
3	XRDY	External Ready	Used by Front Panel: Pulling this line low will cause the processor to enter a WAIT state and allows the status of the normal Ready Line (PRDY) to be examined.
4	V10	Vectored Interrupt Line #0	

BUS DEFINITION

Front Side

No.	SYMBOL	NAME	FUNCTION
5	V11	Vectored Interrupt Line # 1	
6	V12	Vectored Interrupt Line #2	
7	V13	Vectored Interrupt Line #3	
8	V14	Vectored Interrupt Line #4	
9	V15	Vectored Interrupt Line #5	
10	V16	Vectored Interrupt Line #6	
11	V17	Vectored Interrupt Line #7	
12 to 17	UNUSED		
18	$\overline{\text{STATUS DSBL}}$	STATUS DISABLE	Allows the buffers for the 8 status lines to be tri-stated
19	$\overline{\text{CC DSB}}$	COMMAND CONTROL DISABLE	Allows the buffers for the 6 output command/control lines to be tri-stated
20	UNPROT	UNPROTECT	Reserved for input to the memory protect flip-flop on a given memory board
21	SS	SINGLE STEP	Used by Front Panel to disable input buffer while panel drives bidirectional data bus

BUS DEFINITION

Front Side

No.	SYMBOL	NAME	FUNCTION
22	$\overline{\text{ADDR DSBL}}$	ADDRESS DISABLE	Allows the buffers for the 16 address lines to be tri-stated
23	$\overline{\text{DO DSBL}}$	DATA OUT DISABLE	Allows the bidirectional data bus drivers for the 8 data lines to be tri-stated for both input and output data buses
24	Ø2	Phase 2 Clock	
25	Ø1	Phase 1 Clock	
26	PHLDA	Hold Acknowledge	Processor control output signal which appears in response to the HOLD signal; indicates that the data and address bus will go to the high impedance state on the 8080. Note: $\overline{\text{ADDR DSBL}}$ and $\overline{\text{DO DSBL}}$ must be driven to ri-state the system bus
27	PWAIT	WAIT	Processor control output signal which acknowledges that the processor is in a WAIT state
28	PINTE	INTERRUPT ENABLE	Processor control output signal indicating interrupts are enabled: may be set or reset by EI and DI instruction and inhibits interrupts from being accepted by the CPU if it is reset

BUS DEFINITION

Front Side

No.	SYMBOL	NAME	FUNCTION
29	A5	Address Line #5	
30	A4	Address Line #4	
31	A3	Address Line #3	
32	A15	Address Line #15	
33	A12	Address Line #12	
34	A9	Address Line #9	
35	DO	Data Out Line #1	
36	DO0	Data Out Line #0	
37	A10	Address Line #10	
38	DO4	Data Out Line #4	
39	DO5	Data Out Line #5	
40	DO6	Data Out Line #6	
41	D12	Data In Line #2	
42	D13	Data In Line #3	
43	D17	Data In Line #7	
44	SM1	M1	Status output signal that indicates that the processor is in the fetch cycle for the first byte of an instruction
45	SOUT	OUT	Status output signal which indicates that the address bus contains the address of an output device and the data bus will contain the output data when $\overline{PWR}$ is active

BUS DEFINITION

Front Side

No.	SYMBOL	NAME	FUNCTION
46	SNIP	INP	Status output signal which indicates that that the address bus contains the address of an input device and the input data should be placed on the data bus when PDBIN is active
47	SMEMR	MEMR	Status output signal which indicates that the data bus will be used for memory read data
48	SHLTA	HLTA	Status output signal which acknowledges a HALT instruction
49	$\overline{\text{CLOCK}}$	CLOCK	2 MHz clock signal
50	GND	GROUND	

Back Side

No.	SYMBOL	NAME	FUNCTION
51	+8V	+8 volts	Unregulated input to 5v regulators
52	-16V	-16 volts	Negative unregulated voltage
53	$\overline{\text{SSW DSB}}$	SENSE SWITCH DISABLE	Disables the data input buffers so the input from the sense switches may be strobed onto the bi-directional data bus
54	$\overline{\text{EXT CLR}}$	EXTERNAL CLEAR	Clear signal for I/O devices (front panel switch closure to ground)

BUS DEFINITION

Back Side

No.	SYMBOL	NAME	FUNCTION
55	CGND	CHASSIS GROUND	
56 to 67	UNUSED		
68	MWRT	MEMORY WRITE	From the Front Panel indicates that the current data on the Data Out Bus is to be written into the memory location currently on the address bus
69	$\overline{PS}$	PROTECT STATUS	Reserved to indicate the status of the memory protect flip-flop on the memory board currently addressed
70	PROT	PROTECT	Reserved for input to the memory protect flip-flop on the memory board currently addressed
71	RUN	RUN	Indicates that the RUN/STOP flip-flop is set to run on the front panel
72	PRDY	READY	Processor command/ control input that controls the run state of the processor; if the line is pulled low the processor will enter a wait state until the line is released

BUS DEFINITION

Back Side No.	SYMBOL	NAME	FUNCTION
73	$\overline{\text{PINT}}$	INTERRUPT REQUEST	The processor recognizes an interrupt request on this line at the end of the current instruction or while halted. If the processor is in the HOLD state or the Interrupt Enable flip-flop is reset, it will not honor the request
74	$\overline{\text{PHOLD}}$	HOLD	Processor command input signal which requests the processor to enter the HOLD state; allows an external device to gain control of address and data buses as soon as the processor has completed its use of these buses for the current machine cycle
75	$\overline{\text{PRESET}}$	RESET	Processor command input; while activated the content of the program counter is cleared and the instruction register is set to 0
76	PSYNC	SYNC	Processor control output provides a signal to indicate the beginning of each machine cycle
77	$\overline{\text{PWR}}$	WRITE	Processor control output used for memory write or I/O output control; continued next page.

BUS DEFINITION

Back Side

No.	SYMBOL	NAME	FUNCTION
77	$\overline{PWR}$	WRITE	Con't.: data on the data bus is stable while the $\overline{PWR}$ is active
78	PDBIN	DATA BUS IN	Processor control output signal indicates to external circuits that the data bus is in the input mode
79	A0	Address Line #0	
80	A1	Address Line #1	
81	A2	Address Line #2	
82	A6	Address Line #6	
83	A7	Address Line #7	
84	A8	Address Line #8	
85	A13	Address Line #13	
86	A14	Address Line #14	
87	A11	Address Line #11	
88	DO2	Data Out Line #2	
89	DO3	Data Out Line #3	
90	DO7	Data Out LIne #7	
91	D14	Data In Line #4	
92	D15	Data In Line #5	
93	D16	Data In Line #6	
94	D17	Data In Line #1	
95	D10	Data In Line #0	

BUS DEFINITION

Back Side

No.	SYMBOL	NAME	FUNCTION
96	SINTA	INTA	Status output signal to acknowledge signal for INTERRUPT request
97	SWO	WO	Status output signal indicates that the operation in the current machine cycle will be a WRITE memory or output function
98	SSTACK	STACK	Status output signal indicates that the address bus holds the pushdown stack address from the Stack Pointer
99	$\overline{\text{POC}}$	Power-On Clear	
100	GND	GROUND	

Power Supply

POWER SUPPLY

FUNCTIONAL DESCRIPTION

The IMSAI PS-C assembly is a modular unregulated power supply for the IMSAI 8080 System. It provides the basic unregulated +8, +16, and -16 system supply voltages. The PS-C requires a 117 volt AC single-phase line input, and includes a line noise filter. 117 volt terminals, both switched and unswitched, are available for line-powered options such as the ventilating fan and auxiliary power outlets on the back panel. A power switch location is provided on the PS-C, for use when a front panel is not installed in the system.

Physically, the PS-C measures 16.5" x 5.75" x 5.5", (42x15x14 cm), and weighs 16 pounds (7.3 kg).

SPECIFICATIONS - PS-C SUPPLY

Power Requirements:

110-120 volts AC, single phase, 500 watts (maximum)

No load voltages - 117 V input, nominal taps (0, #3):

9.7 volts +18.0 volts -18.0 volts

Current Supplied: at 117 volt input, resistive load:

28.0 amperes at +7.0 volts ripple valley

4.5 amperes at +13.5 volts ripple valley

4.5 amperes at -13.5 volts ripple valley

THEORY OF OPERATION

The PS-C assembly is an unregulated power supply that provides the basic +8, +16, and -16 voltages for the 8080 system. It is comprised of four major component assemblies: line filter, transformer, rectifiers, and filters.

The line filter is a triple PI L-C filter designed to remove high frequency noise present on the A.C. line. This filter attenuates line noise above 1 MHZ in frequency.

The transformer primary is designed for a nominal 117 volt line, and 0, -10%, and +10% taps allow for adjustment to line voltage variations. The transformer secondary is connected as three series winding with a center tap. Four MR 1121 diodes full-wave rectify the +8 volts, while a full-wave bridge of four MR 501 diodes rectify the $\pm$16 volts.

The $\pm$16 volt supplies are each filtered by a 10K uf capacitor to ground, providing +15 average volts a 4.0 amps. The +8 volts is filtered by two 95K uf capacitors to ground, providing 7.3 average volts at the 28 amp rated current.

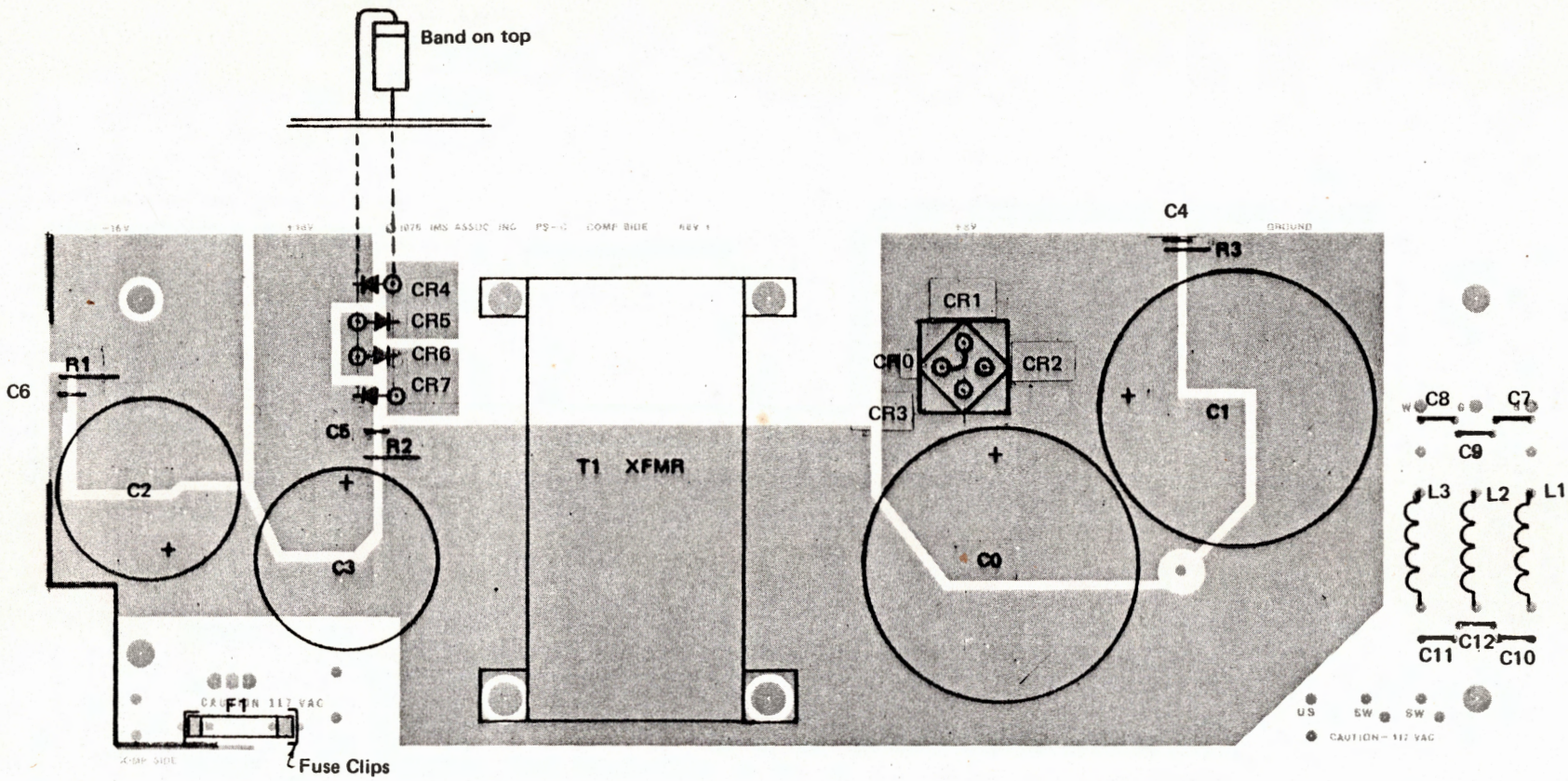
.1 uf capacitors bypass each voltage supply, and bleeder resistors discharge the filter capacitors when power is turned off.

BOARD: PS-C

ITEM	IMSAI PART #	QUANTITY	DESCRIPTION/IDENTIFYING MARKS
Solder	15-0000001	5'	Rosin Core
Heat Sink	16-0100006	1	Thermalloy, Metallic, Modified
Screw	20-3402001	4	6-32x3/8" Phillips Pan Head Machine
Screw	20-4901001	5	8-32x1¼" Binding Head Machine
Screw	20-6901001	4	¼"-20x1½" Binding Head Machine
Screw	20-5201001	8	10-32x½" Binding Head Machine
Nut	21-3120001	4	6-32 Hex Nut
Lockwasher	21-3350001	4	#6 Internal Star Lockwasher
Nut	21-4120001	5	8-32 Steel Nut
Lockwasher	21-4350001	5	#8 Internal Star Lockwasher
Spacer	21-4600002	5	8-32x½" Nylon Threaded
Nut	21-5120001	4	#10 Hex Nut
Lockwasher	21-5320001	4	#10 Split Steel Ring
Lockwasher	21-5350001	8	#10 Internal Star
Nut	21-6120001	4	¼"-20 Steel
Washer	21-6310001	4	¼"-10x1/16" Steel Flat
Lockwasher	21-6320001	4	¼" Steel Split Ring Lockwasher
Washer	21-6390001	4	¼"-10x1/16" Nylon Flat
Spacer	21-6600001	4	¼"-20x½" Nylon Threaded
Wire	22-1014001	48"	White, 14 gauge stranded, insulated
Wire	22-1014002	48"	Black, 14 gauge stranded, insulated
Wire	22-1018001	60"	Orange, 18 gauge stranded, insulated
Wire	22-1020001	60"	Yellow, 18 gauge stranded, insulated
Wire	22-5018001	12"	Orange & Yellow 18 gauge, twisted pair, stranded, insulated
Line Cord	22-6000001	1	Belden 17239
Grommet	24-0600001	1	Strain relief bushing
Transformer	29-0100001	1	4-3751 (or equivalent)

ITEM	IMSAI PART #	QUANTITY	DESCRIPTION/IDENTIFYING MARKS
Inductor	29-0200001	3	8uh, 5 amp
Resistor	30-3470462	1	470 Ohm, ½ watt/yellow, violet, brown
Resistor	30-4100462	2	1K Ohm, ½ watt/brown , black, red
Capacitor	32-2004010	6	.04ufd,500v Disk Ceramic (or .01ufd)
Capacitor	32-2010010	3	.1uf, 30V
Capacitor	32-2510060	2	10Kufd, 25v
Capacitor	32-2595060	2	95Kufd, 15v
Fuse	33-0100001	1	5 Amp.
Fuse Clip	33-0200001	2	102068 Small Fuse Clip
Fan Guard	34-0200001	1	Rotron Fan Guard
Diode	35-1000002	4	MR1121
Diode	35-1000003	4	MR501
PC Board	92-0000024	1	Printed Circuit Board

DIODE MOUNTING



IMS ASSOCIATES INC.
ASSEMBLY DIAGRAM
PS–C REV. 1 5/76

©1976
IMS ASSOCIATES, INC.
SAN LEANDRO, CA.

6/23/76

PS-28
Parts List

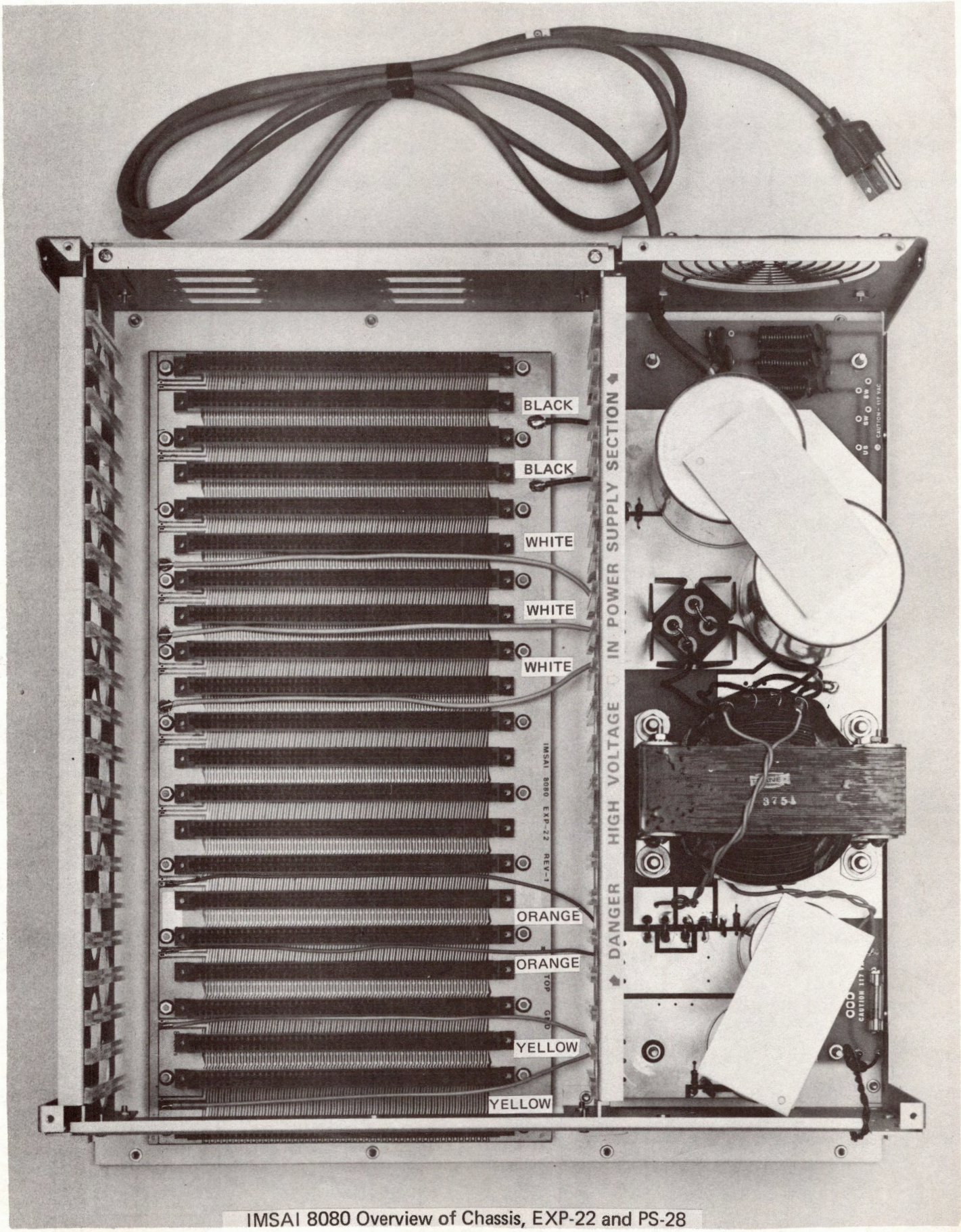
This package contains:

QUANTITY	IMSAI PART #	ID MARKS
5'	15-0000001	rosin core solder
1	16-0100006	thermalloy, metallic, modified heat sink
4	20-3402001	6-32x3/8" Phillips pan head machine screw
5	20-4901001	8-32x1¼" binding head machine screw
4	20-6901001	¼"-20x1½" binding head machine screw
8	20-5201001	10-32x¼" binding head machine screw
4	21-3120001	6-32 hex nut
4	21-3350001	#6 internal star lockwasher
5	21-4120001	8-32 steel nut
5	21-4350001	#8 internal star lockwasher
3	21-4600002	8-32x½" nylon threaded spacer
4	21-5120001	#10 hex nut
4	21-5320001	#10 steel split ring lockwashers
8	21-5350001	washer, internal star
4	21-6120001	¼"-20 steel nut
4	21-6310001	¼"-10x1/16" Steel Flat
4	21-6320001	¼" steel split ring lockwasher
4	21-6390001	¼"-10x1/16" Nylon Flat
4	21-6600001	¼"-20x½" nylon threaded spacer
48"	22-1014001	White, 14 gauge stranded, insulated wire
48"	22-1014002	Black 14 gauge stranded, insulated wire
60"	22-1018001	Orange, 18 gauge stranded, insulated wire
60"	22-1018002	Yellow, 18 gauge stranded, insulated wire
12"	22-5018001	Orange and Yellow, 18 gauge twisted pair, stranded, insulated wire
1	22-6000001	Belden 17239 line cord
1	24-0600001	strain relief bushing grommet
1	29-0100001	4-3751 transformer (or equivalent)
3	29-0200001	8uh, 5 amp inductor
1	30-3470462	470 Ohm, ½ watt resistor (yellow/violet/brown)
2	30-4100462	1K Ohm, ½ watt resistor (brown/black/red)
8	32-2004010	.04 ufd, 500 V disk ceramic capacitor or .01 ufd
3	32-2010010	.1uf 30v capacitor
2	32-2510060	10K ufd, 25v capacitor
2	32-2595060	95K ufd, 15v capacitor
1	33-0100001	5 amp fuse
2	33-0200001	102068 small fuse clip
1	34-0200001	guard, Rotron fan
4	35-1000002	MR1121 diode
4	35-1000003	MR501 diode
1	92-0000024	printed circuit board

This is a revised parts list and may differ from your manual in some respects. Please use this list for parts check off.

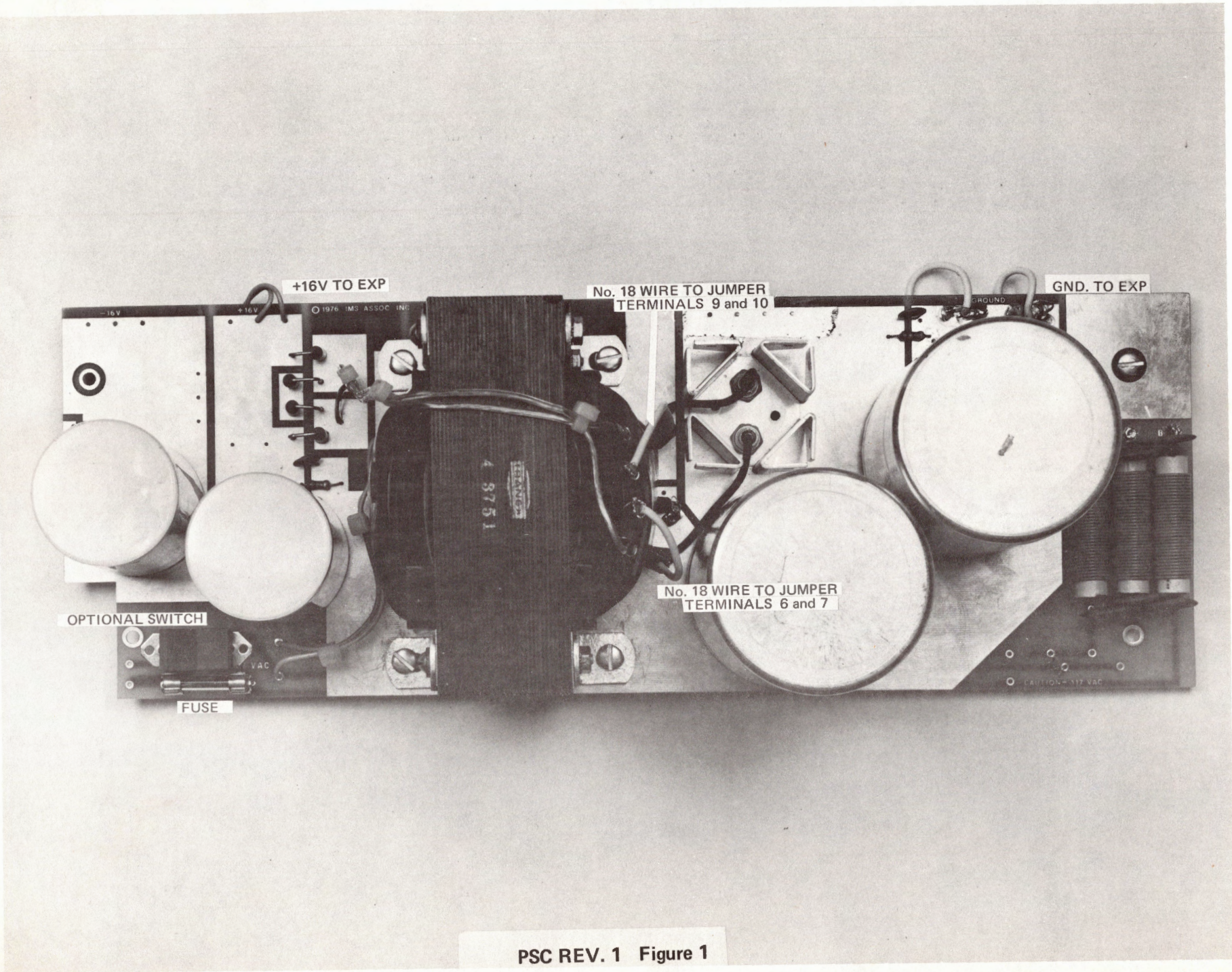
8/12/76

11-22

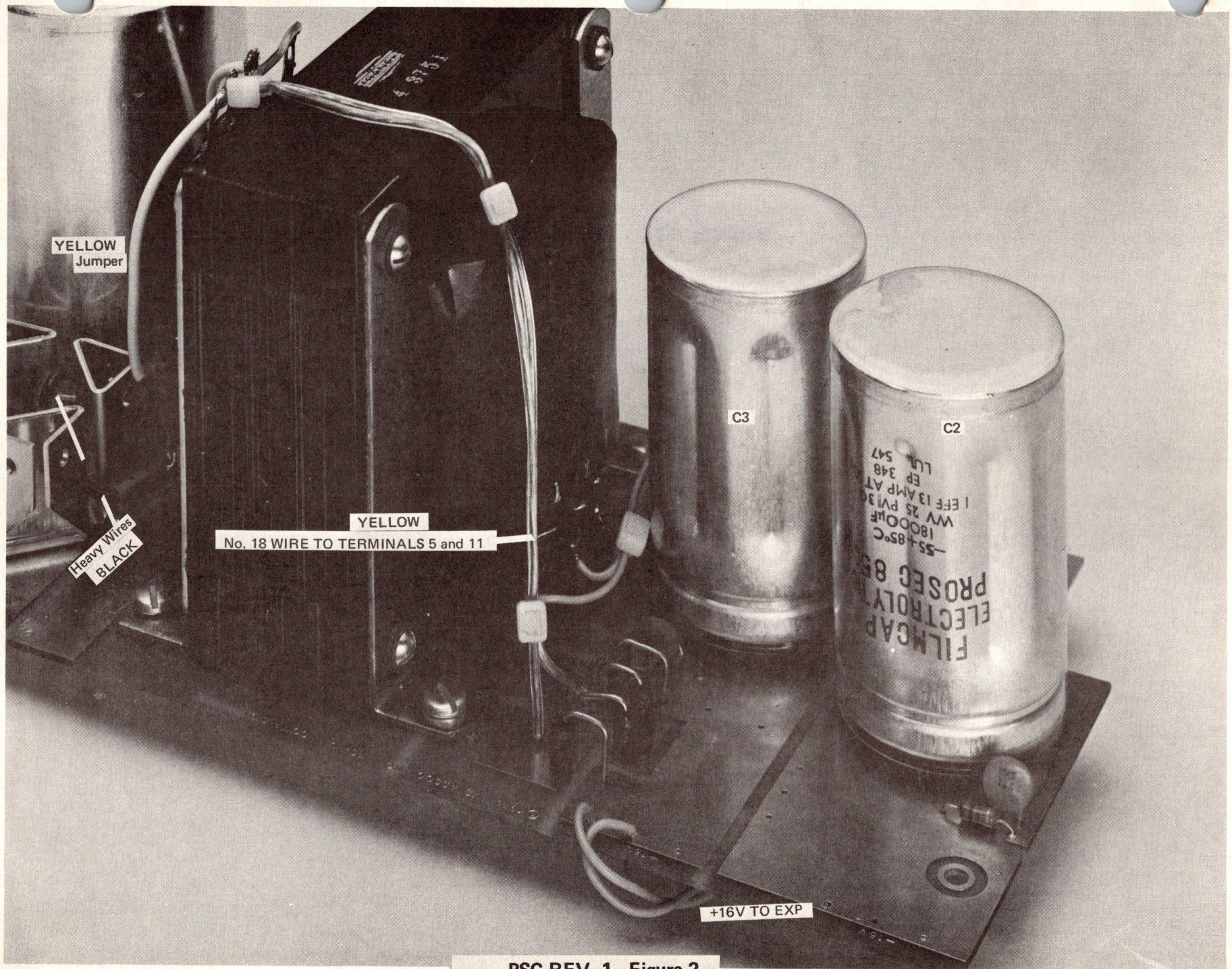


IMSAI 8080 Overview of Chassis, EXP-22 and PS-28

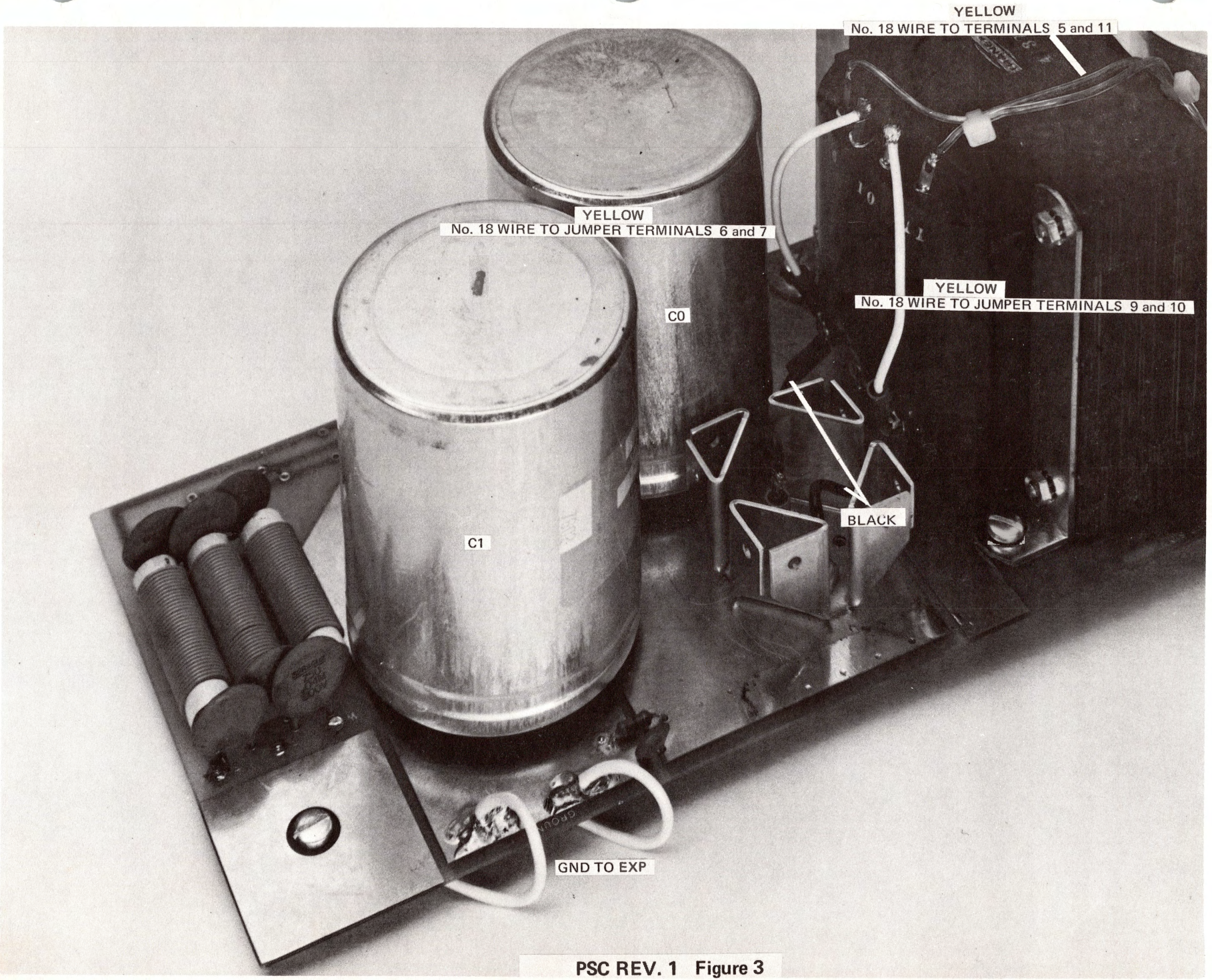
PS–C OVERALL VIEW



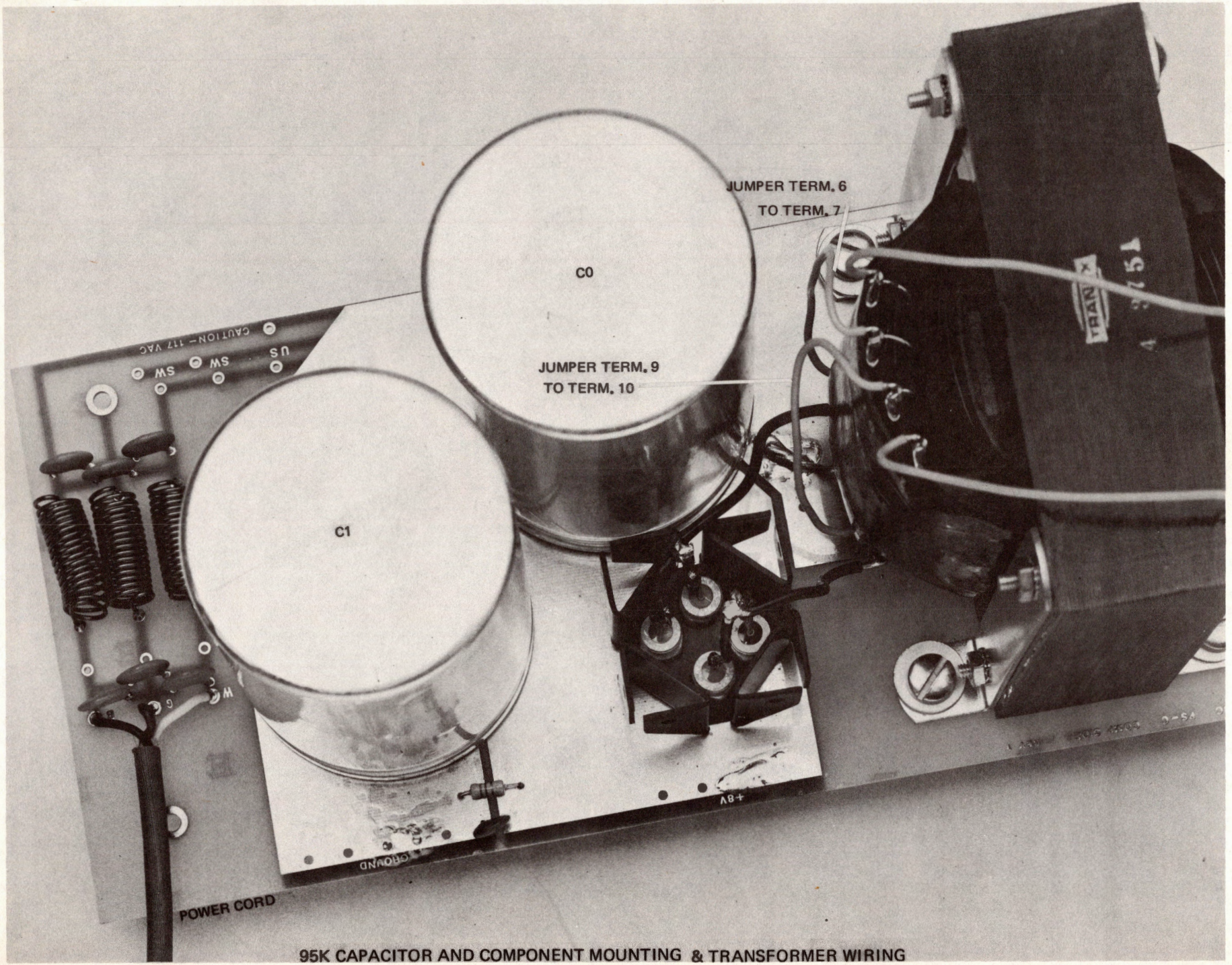
PSC REV. 1 Figure 1



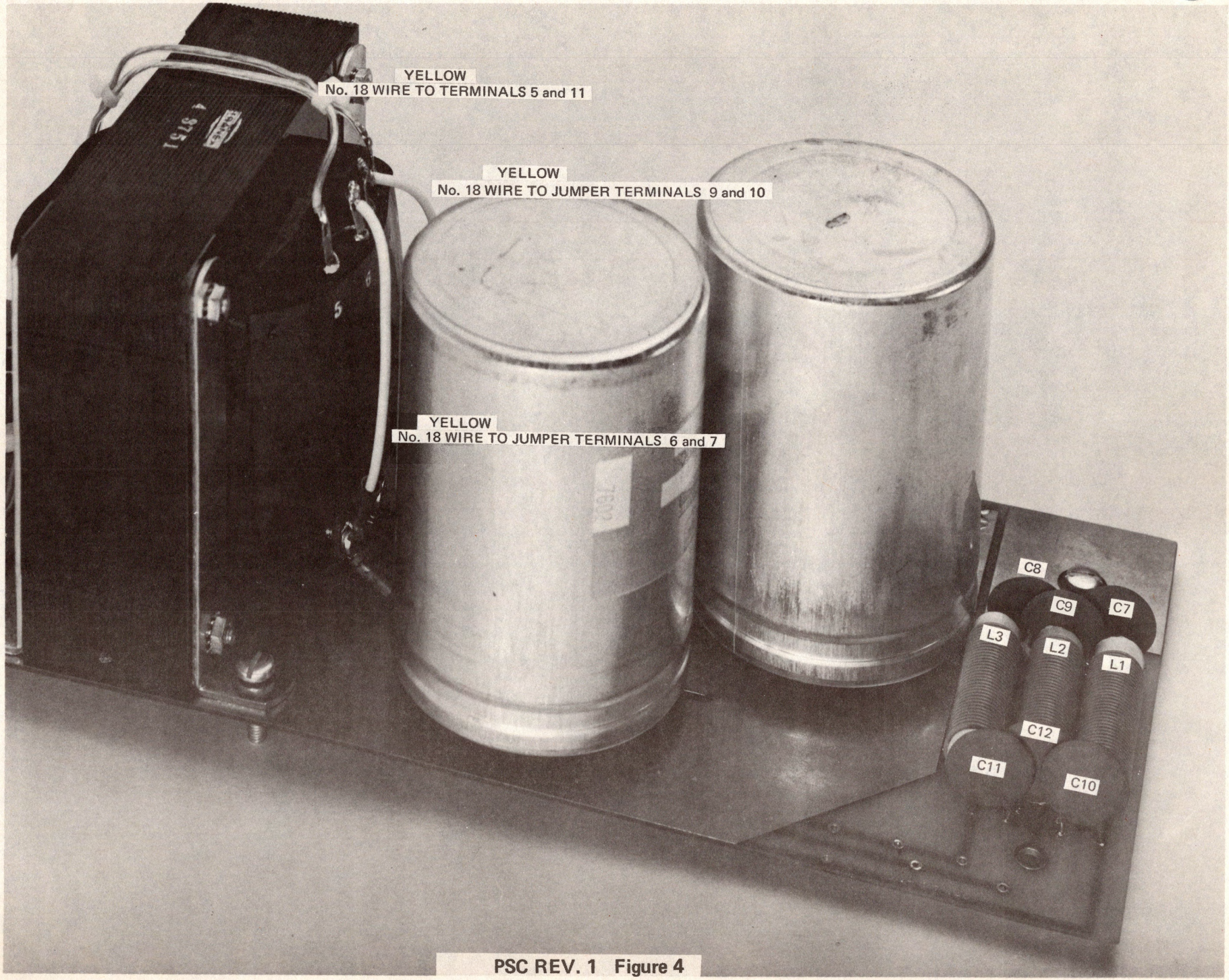
PSC REV. 1 Figure 2



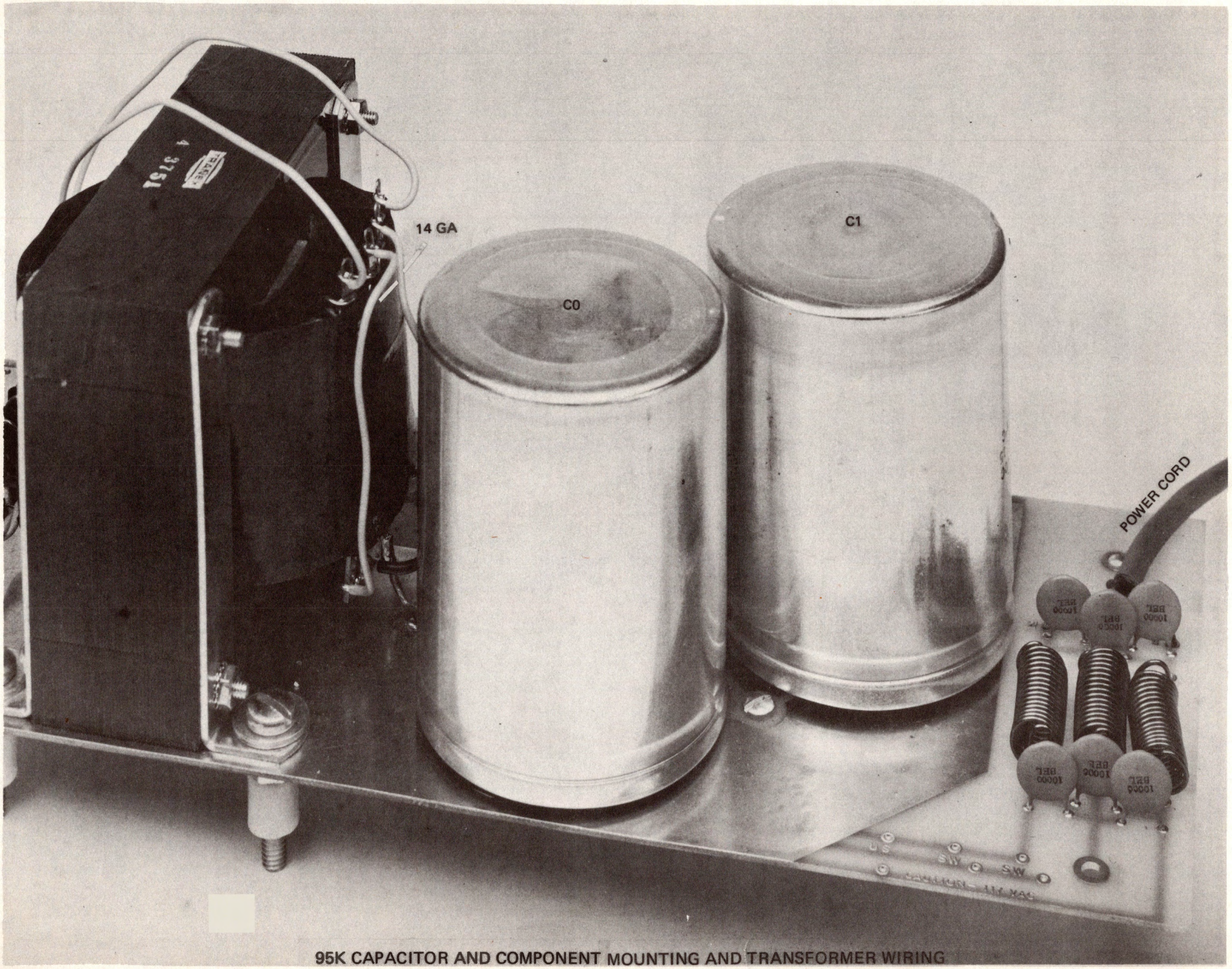
PSC REV. 1 Figure 3



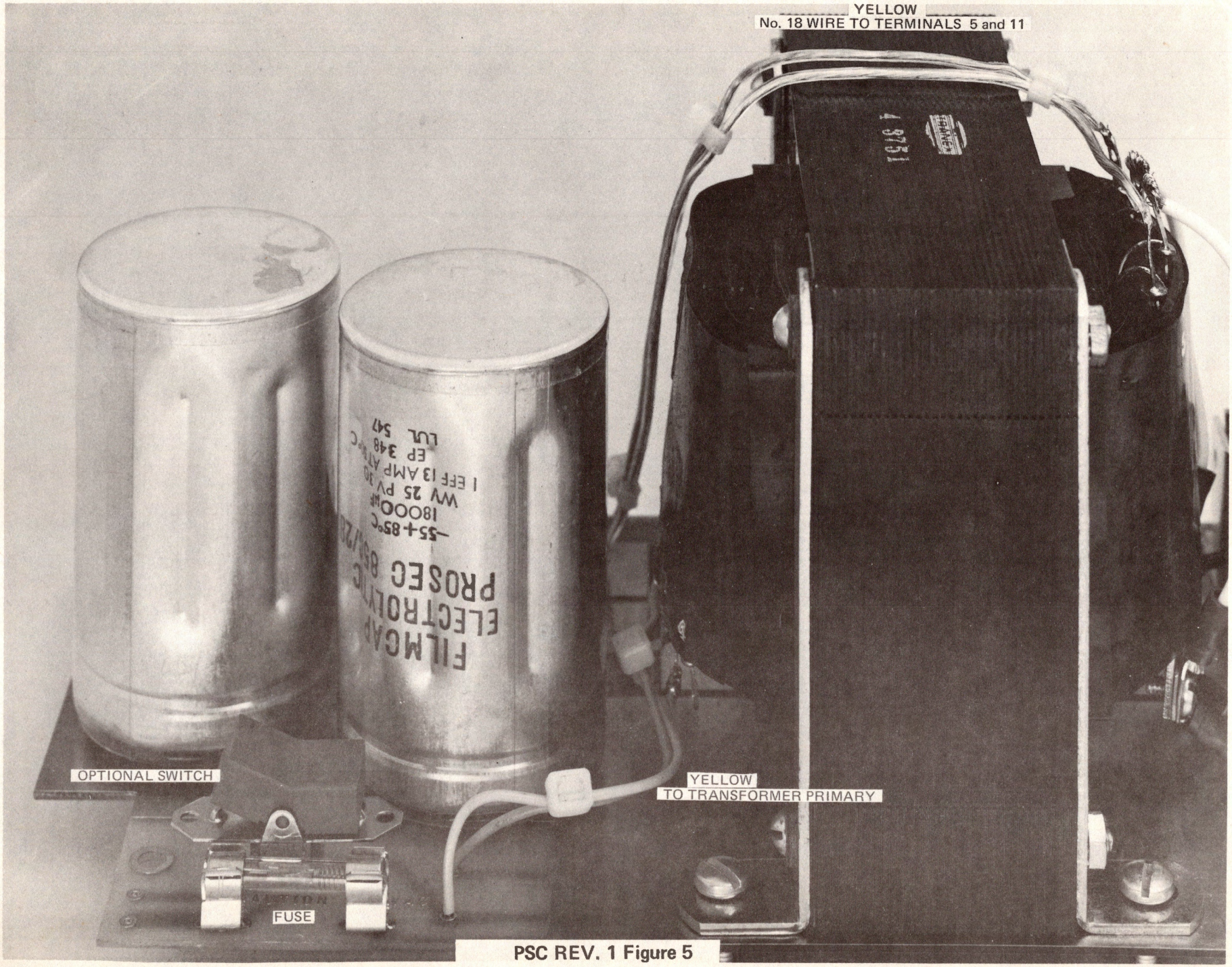
95K CAPACITOR AND COMPONENT MOUNTING & TRANSFORMER WIRING



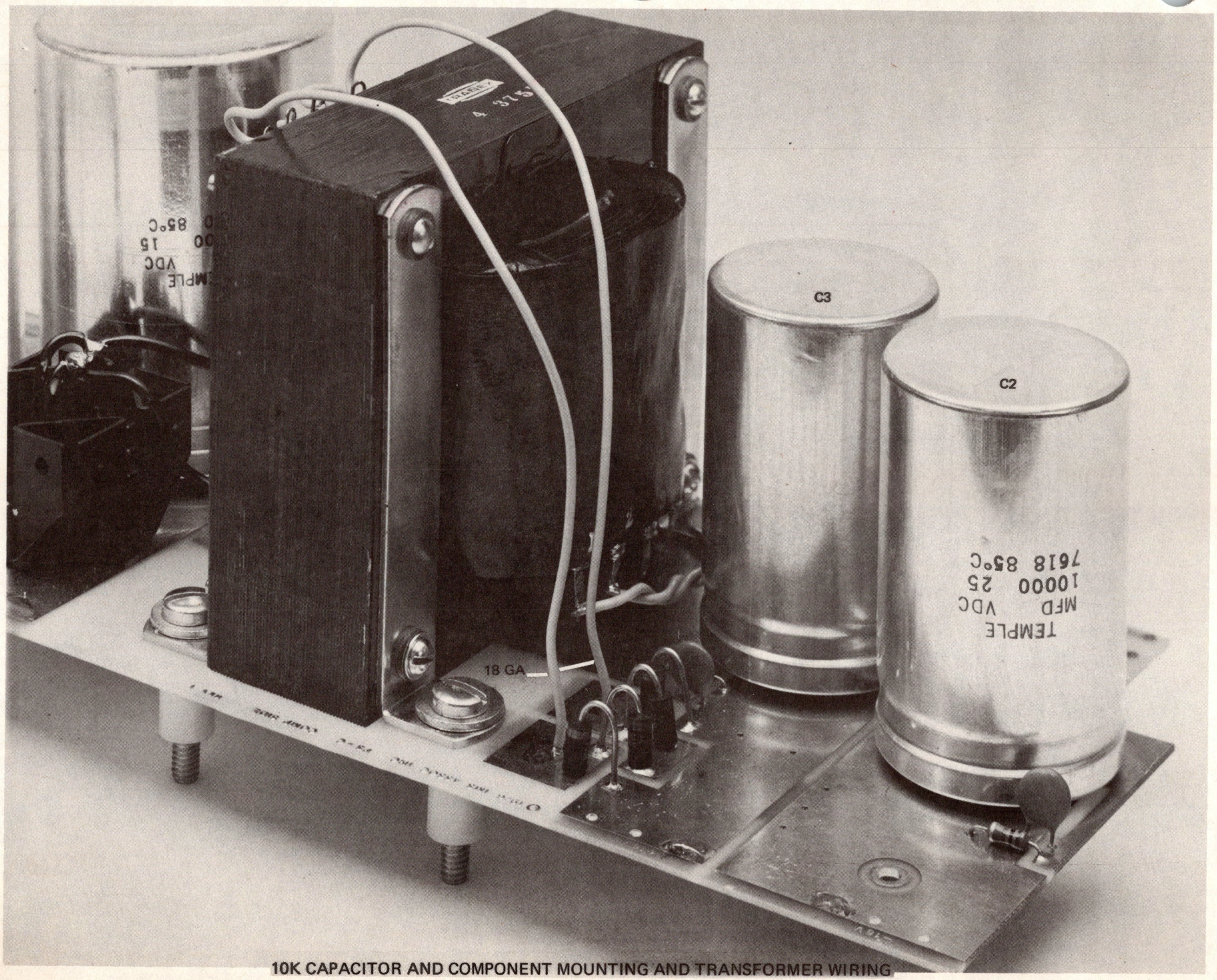
PSC REV. 1 Figure 4



95K CAPACITOR AND COMPONENT MOUNTING AND TRANSFORMER WIRING



PSC REV. 1 Figure 5



10K CAPACITOR AND COMPONENT MOUNTING AND TRANSFORMER WIRING

When to mount Hornets

POWER SUPPLY

ASSEMBLY INSTRUCTIONS

1) Unpack your board and check all parts against the parts lists enclosed in the package.

COMPONENT INSTALLATION

2) Insert and solder each of the two 1K ohm 1/2 watt resistors (brown/black/red) R1 and R2 at locations as shown on the Assembly Diagram.

3) Insert and solder the one 470 ohm 1/2 watt resistor (yellow/violet/brown) R3 at location shown on the Assembly Diagram.

4) Insert and solder each of the three .1 uf capacitors C5, C6 and C4 at locations indicated on the Assembly Diagram.

5) Next bend each of the cathode leads on each on the four rectifier diodes CR4, CR5, CR6, and CR7 as shown in Figure 2. Insert the anode end of the diodes down as shown in Figure 2 and solder. NOTE: See Assembly Diagram for diode mounting position.

6) Insert and solder each of the six .04 uf capacitors C7 through C12 in the locations shown on the Assembly Diagram.

7) Insert and solder each of the three AC filter inductors L1, L2 and L3 in locations as shown on the Assembly Diagram.

8) Insert and solder each of the two fuse clips in locations as shown on the Assembly Diagram. See Figure 5 for clarification. Snap in fuse.

TRANSFORMER WIRING

Put Xformer next to P.C. Bd.

9) Solder the yellow wire to the transformer lugs numbers 1 and 2 for AC. Wires should be twisted together (not shown as twisted in photographs). Solder twisted pair ends to PSC board as shown in Figure 5.

Yellow Length Wire?

Cannot Do?

10) Solder the two yellow wires on the secondary side of the transformer to lugs numbers 6 and 10. Attach the other ends to terminal numbers 7 and 9 respectively.

Length Wire?

Jumper 6 & 7 and 9 & 10

(Do not Solder)

Drawing Shows (7-6 & 9-10)?

Length Wire?

11) Take the two black wires (#14 or larger) and attach to the number 7 and 9 terminals.

12) Next connect the two yellow #18 wires to terminal numbers 5 and 11. Run the wires across the transformer to the anode pads of CR4 and CR7. Solder.

Length Wire?

13) Temporarily install the transformer with ¼" hardware. Terminals 1,2,3, and 4 should face towards the fuse end of the PSC board.

NOTE: Scrape or sand lugs 7, 8, and 9 of transformer to facilitate solder connections of #14 wires.

14) Take a short wire and solder to terminal 8 on the transformer.

15) Solder the other end of the short wire used above (#14 or larger) to the ground trace directly below terminal number 8.

HEAT SINK INSTALLATION

NOTE: The heat sink included in your kit is black anodized aluminum (silver heat sink shown in pictures). Keep all wiring as short as possible, an extra two inches of #14 wire will reduce the current capacity of the Power Supply.

16) Insert the four 1121 rectifier diodes CR0 through CR3 through the heat sink (only two are shown in Figures 1 and 3). Solder a wire between the anodes of CR0 and CR1 and solder a wire between the anodes of CR2 and CR3. The wire used should be #14 or larger (the black wire).

17) Solder the black wire from terminal 7 to wire from terminal 6 to CR0 and CR1.

18) Repeat above (#17) procedure from black wire from terminal 9 to CR2 and CR3.

19) Install and bolt heat sink (and diodes) onto the PSC board.

NOTE: WARNING! OBSERVE POLARITY

The large capacitor will be destroyed if power is applied while they are installed backwards.

On the two large capacitors C0 and C1, the negative side of the capacitor bolts to the DC ground plance of the PSC board. The positive end of capacitors C0 and C1 bolts to the unregulated 8 volts plane of the PSC board.

20) Place lockwashers on four 10-32x3/8" screws. insert them from the underside of the board and mount capacitors C0 and C1.

21) In a similar manner mount C3 with the negative terminal bolted to the ground plane and positive terminal bolted to the +16 volt plane.

22) To install capacitor C2, bolt the positive terminal to the DC ground plane and the negative terminal to the negative (-16 volt) plane.

23) Attach the capacitor brace plates to the bases of the large capacitors with the adhesive backed foam tape located on one side of the brace plate.

SEE MAINFRAME ASSEMBLY SECTION TO INSTALL POWER SUPPLY IN THE CHASSIS AND CONNECT TO THE MOTHER BOARD.

USER GUIDE

The PS-C User's only significant option is the selection of a transformer primary tap. The transformer provides 0 (terminal #3), +10% (terminal #4), and -10% (terminal #2) taps, and the user should select a tap appropriate to his line voltage and load current requirements.

As the PS-C is an unregulated supply, the supply voltages are dependent on the load conditions. The user may adjust his loaded voltage by picking an appropriate primary tap, but should be careful that the no load voltages do not exceed +11, +18, and -18 volts, these maximums are selected so that the power dissipated in the system's voltage regulators and zener diodes does not exceed the device ratings. Similary, the user should not allow the +8 supply to fail below 7.0 volts, the point at which the 7805 regulators cease to regulate.

The ultimate current at 7.0 volts ripple valley increases when the -10% tap is selected, and users who expect to draw more than 15 amperes continuously should connect to this tap. For example, a user with three cards in his system, drawing approximately 4 amperes should connect to the standard tap, while for a 25 ampere (full card cage) load the -10% tap should be connected.

The +10% tap should be selected if the line voltage is high enough that the no load voltages are above the maximums.

Large currents require extremely low resistance paths from the power supply to the motherboard. It is suggested that #14 wire in multiple lengths be used to connect the power supply to the motherboard, and that all wires be only as long as necessary. Special care is required to insure low resistance solder connections; the +8 and ground leads are especially critical in this regard. Any significant IR loss in the supply wiring reduces the power available at the motherboard.

The power switch leads may be connected to the pads provided, or the user may mount a switch directly on the PS-C board. Unswitched (marked US) and switched (marked SW) AC pads allow the connection of fans or external equipment.

CP-A

OPTIONAL MODIFICATION OF CP-A REVISION 4 OR EARLIER CP-A BOARDS TO CHANGE POWER SWITCH TO WRITE PROTECT/UNPROTECT SWITCH FOR USE WITH RAM 4A BOARDS.

REMOVE CP-A FROM CHASSIS

A. Remove AC leads from pads A & B on CP-A, route to miniature toggle switch (e.g., C&K type 7101) mounted in ¼" hole (provided) in rear of chassis. Connect to center and bottom terminals of switch.

B. Carefully suck solder away from terminals of Power switch on CP-A using a solder sucker or pieces of copper braid. Use enough heat to melt solder, but do not overheat board. Unscrew the Power switch from the bracket and heat all 3 terminals simultaneously. (Use 3 irons, or "timeshare" one--moving between the terminals.) When all 3 terminals are hot enough, the switch will easily lift out. DO NOT PULL. Pulling will damage the pads.

C. Cut the following traces (refer to diagram):

Between center and upper AC Power switch terminal (front side),
ground lead going to HOLD light (back side),
trace to resistor on HOLD light (after feed through) (backside),
trace to Mother board connector pin 20 (after feed through) (backside),
trace to Mother board connector pin 70 (after feed through) (front side).

D. Re-install a 3-position momentary switch in old AC Power switch position.

E. Install two 470 Ohm, ¼ watt resistors between ground and the heavy traces from the switch (or to U24 pins 12 & 14).

F. Install the following jumpers:

From +5 to resistor from old HOLD light (other end than LED)

From U22 pin 6 to switch center terminal

From pin 15 to pin 1 on U24

From the heavy trace (to the bottom AC Power switch terminal) to U24 pin 14

From the heavy trace (to the top AC power switch terminal) to U24 pin 12

From U24 pin 13 to the pad connected to pin 70

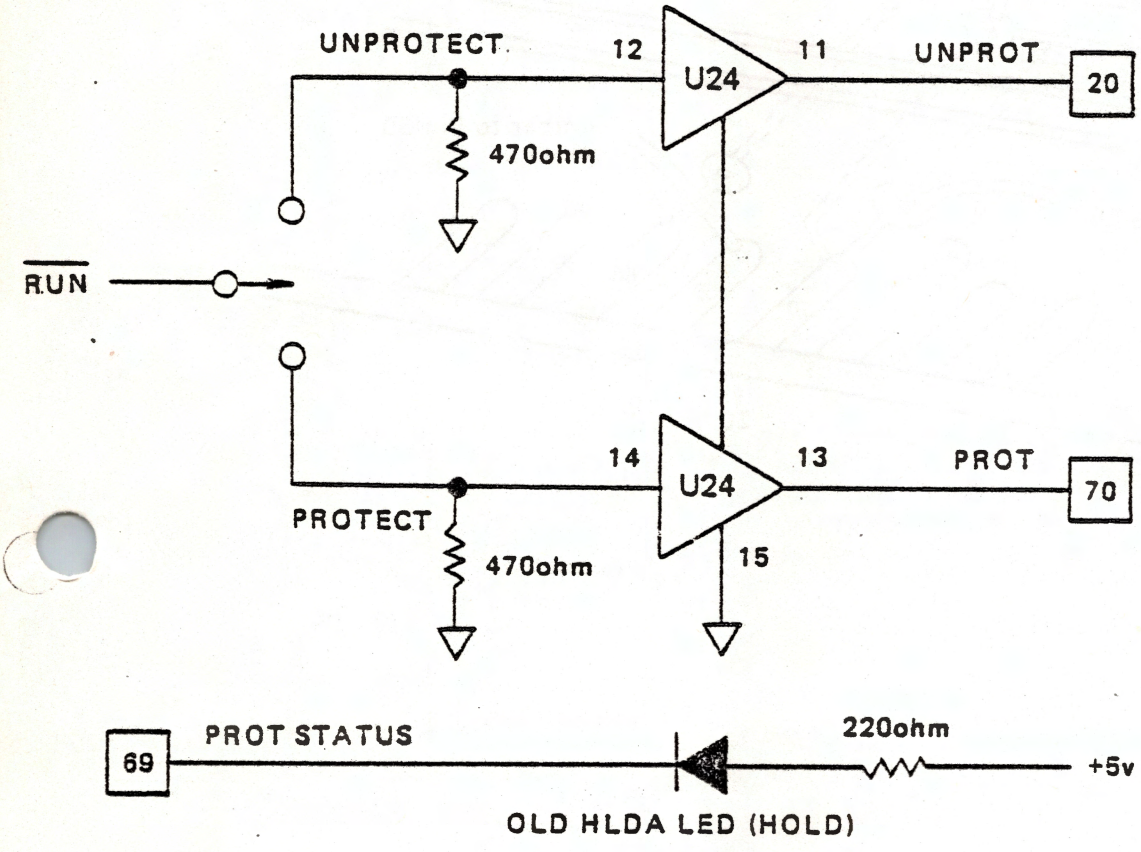
From U24 pin 11 to the pad connected to pin 20

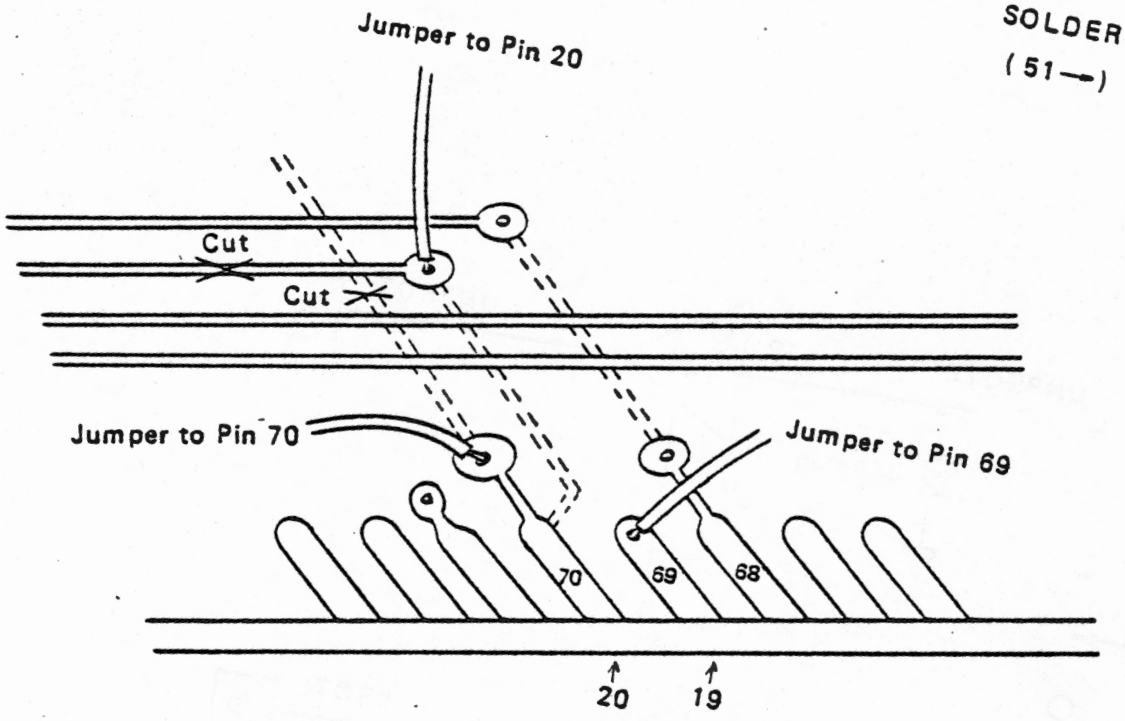
From the bottom terminal of the HOLD light (cut from ground) to pin 69 (solder to the top 1/16" of finger only)

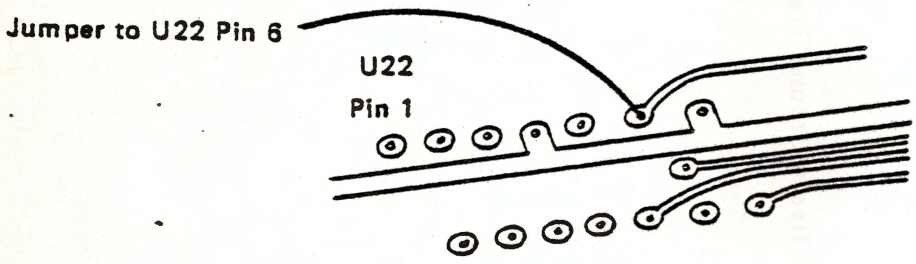
Insert labels to change HOLD (for light) to MEM. PROTECT and to change POWER ON/POWER OFF to PROTECT/UNPROTECT.

Re-assemble CP-A to chassis

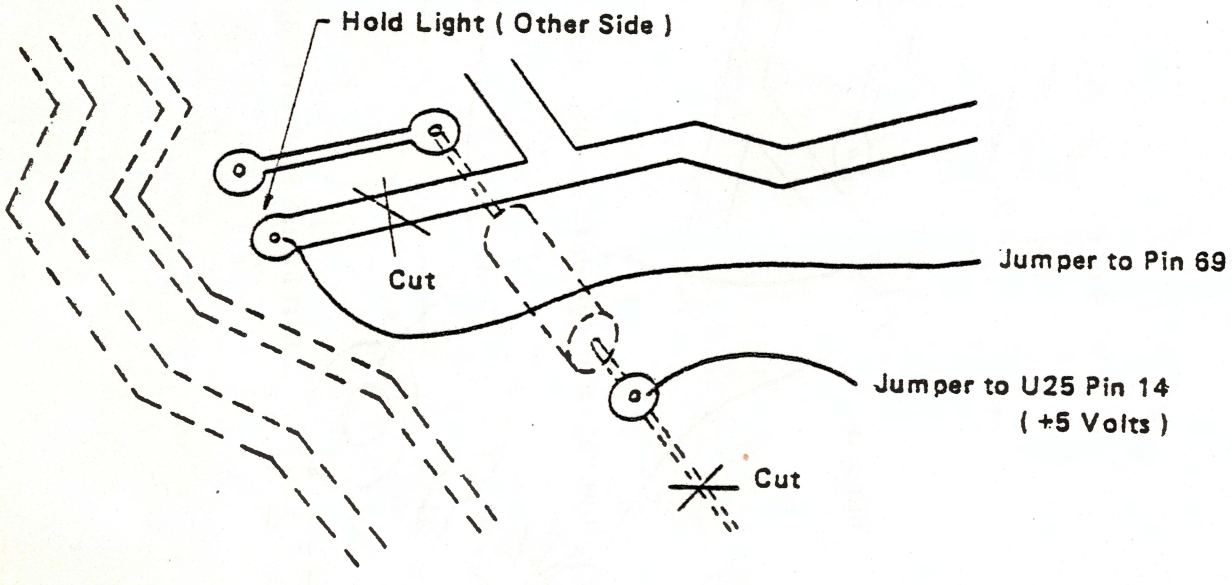
The right-hand switch now serves to change the protect status of the currently addressed block of memory when the machine is not in run mode. The LED which previously showed HOLD status now is lit when the currently addressed block of memory is protected.


UNPROTECT
12
U24
11
UNPROT
20
470ohm
RUN
14
U24
13
PROT
70
PROTECT
470ohm
15
69
PROT STATUS
220ohm
+5v
OLD HLDA LED (HOLD)

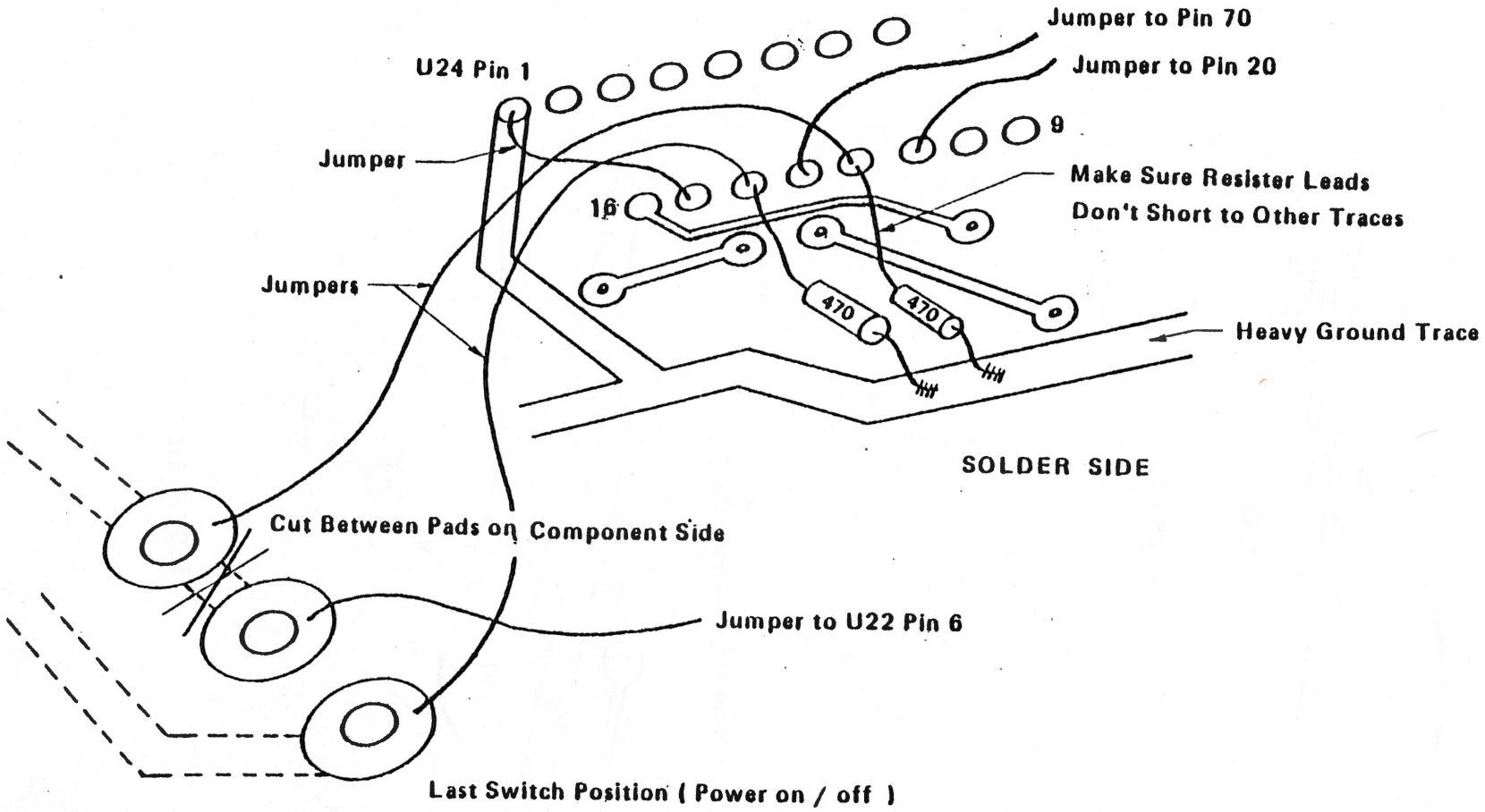

Jumper to Pin 20
SOLDER SIDE
(51→)
Cut
Cut
Jumper to Pin 70
Jumper to Pin 69
70
69
68
20
19

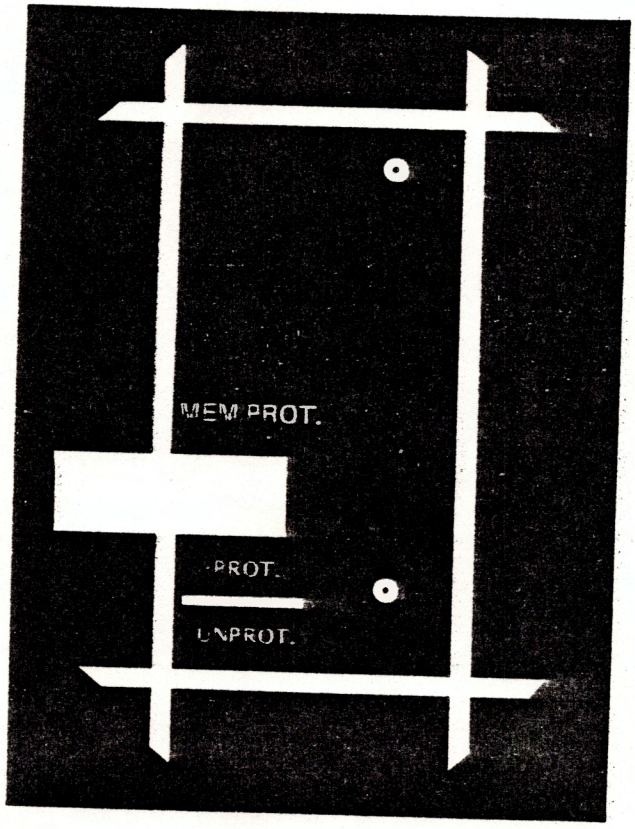


SOLDER SIDE



SOLDER SIDE


Jumper to Pin 70
Jumper to Pin 20
U24 Pin 1
9
Jumper
16
Make Sure Resister Leads
Don't Short to Other Traces
Jumpers
470
470
Heavy Ground Trace
SOLDER SIDE
Cut Between Pads on Component Side
Jumper to U22 Pin 6
Last Switch Position (Power on / off)


MEM PROT.
PROT.
UNPROT.

CP-A

FUNCTIONAL DESCRIPTION

The CP-A board is the operator's panel for the IMSAI 8080 System. It includes operator switches, indicator lights and all logic necessary to operate the IMSAI 8080 System.

The panel is completely self-contained and plugs into the back plane's 100 pin edge connector. With this design it is not necessary to mount the CP-A at the front of the cabinet. Instead, the board can be plugged (via an extender card) into any available slot in the back plane.*

A full set of 16 address switches and 6 control function switches accept operator control and input. LED indicators are provided for the 16 bit address bus, the 8 bit data bus, the 8 bit status byte (control indicators for INTERRUPTS, ENABLED, RUN, WAIT and 8 bits of programmed output.

The CP-A board contains the logic necessary to drive the 8 programmed output indicators and the logic needed to read an 8-bit input byte from the high-order address switches. The DATA BUS indicators are run from the bi-directional portion of the data bus (via a flat cable to the MPU board) and show data either being read or written by the 8080 processor.

The indicators on the panel are wide-angle-view light emitting diodes mounted behind a contrast-enhancing acrylic panel assembly. All indicators and switches are explicitly marked. The photographically produced labels are very clear, protected by clear acrylic, and can never wear off. Bit positions are numbered, and binary bit values are labeled for both hexadecimal

*The switches are included on the front panel whether it is mounted in the front of the panel or not.

and octal formats. Special labels may be easily inserted to identify special functions for the programmed output port. Switches on the panel are high-quality paddle switches, and are color-coded for easy and error free use.

For situations in which it is not desired to locate the operator's panel at the cabinet front (such as use of the IMSAI 8080 as a dedicated controller), the CP-A front panel may be inserted (via extender card) into any back plane slot. In this arrangement, programs may be easily tested and debugged without time-consuming mounting and un-mounting of the front panel. For these applications, the front slot of the machine can be reserved for the parallel I/O board with its LED indicators showing through the front panel mask.

THEORY OF OPERATIONS

The CP-A front panel assembly provides machine status indicators, user controlled switches, and control functions to the IMSAI 8080 operator. The CP-A board communicates with the MPU-A microprocessor and other boards through the 8080 back plane and, additionally, connects (via 16 conductor flat cable) to the bi-directional data bus of the 8080 microprocessor.

The CP-A panel uses 44 Light Emitting Diodes as front panel indicators. Many of these indicators directly correspond to signal levels on the IMSAI 8080 back plane, and are driven directly from the bus with no intervening logic. Indicators in this group are the 16 Address Bus LED's, the 8 STATUS byte LED's, the INTERRUPT ENABLED LED, the WAIT LED, and the HOLD LED.

The 8080 microprocessor chips bi-directional data bus levels (provided by a 16-conductor cable) are displayed on the DATA bus indicators via the 74LS04 (low power schottky hex inverter) sections. Also driven from the bi-directional bus is the 8212 8 bit latch used to drive the PROGRAMMED OUTPUT indicators. The RUN indicator is driven directly from the run/stop flip-flop (74107) on the CP-A Board.

The 16 ADDRESS-PROGRAMMED INPUT and ADDRESS-DATA switches allow the operator to place desired value (program, data, addresses) on the 8080's bi-directional bus.

As shown on the schematic, these switches connect 7405 (open collector) inverters to the bus in a wired- AND configuration.

Pullup resistors on the MPU Board ensure that the bus levels are all high unless any inverter on any one of the bus lines goes low. Thus, if an inverter goes low, (this condition will be discussed shortly) the address switch can be used to put either a high or low value on that line.

The function switches provide the operator with direct control of the microprocessor. The RUN/STOP switch controls the X-READY line via the RUN/STOP flip-flop.

If the switch is set to RUN, on the next falling edge of the Phase II clock, the RUN and X-READY lines are set high. If the switch is set to STOP, the high STOP value and the Phase II clock are NANDed (U16) and this value NANDed with the DATA OUT 5 bit (fetch/status) and the PROCESSOR SYNC line.

Thus, when the processor is fetching a new instruction, the RUN/STOP flip-flop will be reset, the processor X-READY line goes low, and the processor stops.

Several CP-A function switches operate by providing the 8080 with an instruction, executing the instruction, and then stopping the processor on the next cycle. The open collector 7405's and support gating put these instructions on the 8080's bi-directional bus. The EXAMINE function uses a jump instruction (hex C3) followed by two bytes of the address selected on the front panel switches.

This operation causes the processor to jump to the selected address and, then, the processor is stopped during the next cycle. When stopped, the processor was reading the selected byte from memory as if it were going to execute it. Therefore, the processor stops with the desired address displayed on the address bus and the contents of that address is displayed on the data bus.

If the RUN switch is operated at this time, the processor will continue to pull the selected byte from memory and execute it.

The EXAMINE NEXT and DEPOSIT NEXT switches use similar schemes and the NO-OP (hex 00 or octal 000) instruction to increment the address.

Much of the remaining logic of the CP-A is used to sequence these commands to provide the desired functions. The RUN/STOP flip-flop line, the SINGLE STEP line, the EXAMINE line, and the EXAMINE NEXT line are all input to an OR-gate controlling the X-READY line. (The X-READY line must be high for the processor to run. Its

function is identical to the P-READY line used by the memory and I/O boards. The X-READY line is reserved for use of the front panel to avoid conflicts of two gates driving the same backplane line). During each of these functions, the processor is permitted to execute an instruction, and then is stopped in the next cycle in a manner similar to the RUN/STOP flip-flop cycle described earlier.

For the SINGLE STEP function, a one-shot, triggered by the SINGLE-STEP switch, is used to produce a pulse and the trailing edge of that pulse is used to set a flip-flop which controls the SINGLE STEP line. This permits the processor to execute the present instruction. The SINGLE STEP flip-flop is reset by the occurrence of the sync pulse on the following instruction, thus causing the SINGLE STEP level to be removed and the processor to stop on the following cycle.

The EXAMINE-NEXT flip-flop is similarly controlled by the leading edge of a pulse from a one-shot driven by either the DEPOSIT NEXT or EXAMINE NEXT switch. The output of the flip-flop is used both to put the NO-OP (hex 00 or octal 000) onto the bi-directional data bus, and also to provide the READY signal so that the processor will execute the instruction. It is reset by the sync pulse on the following cycle, thus stopping the processor again.

The EXAMINE function involves a 4-step sequence produced by two flip-flops arranged as a counter. The pulse produced by the EXAMINE switch's one-shot starts the counter and on the first count, the jump instruction is inserted on the data bus. On successive counts of the two bit counter, the lower and upper address byte are inserted on the data bus in turn, and on the 4th count (that is, when the counter is back to 0), the processor is again stopped by the removal of the READY line. Thus, the EXAMINE logic provides the processor with the jump instruction and the two address bytes that the processor expects after a jump instruction and

stops the processor during the fetch of the designated memory byte.

Similarly, the DEPOSIT switch, when operated, produces a pulse from the DEPOSIT one-shot which is buffered to the MEMORY WRITE line on the backplane. The leading edge of this pulse also starts a second one-shot with a much longer period which puts the data from the data switches on to the data bus for the duration of the longer pulse. The DEPOSIT one-shots are triggered either by the operation of the DEPOSIT switch or by the trailing edge of the DEPOSIT NEXT one-shot so that the DEPOSIT function will operate at the end of the EXAMINE NEXT cycle.

The 7427 gate in U15.5 is used to insure that during the time the front panel is inserting any information on the bi-directional data bus, the MPU-A board's bi-directional data bus driver is not also trying to drive the bus at the same time.

The inputs to this gate are the DATA-ON line, the EXAMINE NEXT line and the EXAMINE line. These are the three functions during which the front panel is transferring data or instructions to the bus.

The inputs to the 7405 open-collector inverter bus drivers are the lines NO-OP, C3, HAD, and LAD. These levels are ANDed with the PDBIN signal so that the information appears on the bus during the time the processor is expecting to see it there.

The input port from the high order address switches is implemented simply by decoding the address FF and ANDing it with the DBIN signal so that switch values appear on the data bus during the time that the processor is expecting information from the port FF.

The same address decode signal is ANDed with the STATUS OUT line to enable the 8212 8 bit latch which drives the PROGRAMMED OUTPUT indicators. The information on the bi-directional data bus is then latched onto the output port at the time of the processor write strobe.

The STATUS WORD DISABLE line (SSWDSB, Pin 53 backplane) is gated to insure that no conflicts are created between the bi-directional bus drivers on the MPU and CP-A boards. This signal is controlled by the same gating that places the high order address switch values on the data bus for a front panel (address hex FF) read.

The STATUS WORD DISABLE line, Pin 53 in the backplane, is also run by the signal which puts the high order address switches onto the data bus for the port FF read instruction so that the bi-directional data bus is not being driven by the bi-directional drivers on the MPU board at the same time that the front panel is inserting the switch information on the data bus.

The RESET switch directly grounds the RESET line on the backplane which is detected by the MPU board and processed to form a RESET pulse which re-appears on the backplane as a Power On Clear.

When the RESET switch is thrown to EXTERNAL CLEAR, the switch directly grounds the EXTERNAL SWITCH line on the backplane. There is a diode between the RESET line and the EXTERNAL CLEAR line so that during a reset operation an EXTERNAL CLEAR is also generated.

CP-A

This package contains:

QUANTITY	IMSAI PART #	IDENTIFICATION
10'	15-0000001	Solder
1	16-0100002	Heat Sink
22	20-2203001	Screw, 4x ¼", sheet metal, self-tapping, slotted hex head integral washer
2	20-3203001	# 6x¼" Self Tapping Sheet Metal Screws
1	20-3302001	Screw,Machine, 6-32 x 5/16" Phillips Pan Head
8	20-3916002	Screw, machine, 6-32 x 1¼", Button Head, Allen, Black Oxide
1	21-3120001	Nut, Hex, 6-32
1	21-3350001	Lockwasher, Internal Star, #6
8	21-3600001	Spacer, Nylon, #6 x ¼"
8	21-3600002	Spacer, Nylon #6 x 7/16"
8	26-1500001	Switch, Paddle, Blue (ONO)
8	26-1500002	Switch, Paddle, Red (ONO)
2	26-1500003	Switch, Paddle, Red (MFM)
3	26-1500004	Switch, Paddle, Blue (MFM)
1	26-1600001	Switch, Rocker, Red (ONO)
44	30-3220362	Resistor, 220 OHM ¼ W 5% Resistor/Red, Red, Brown/
9	30-4100362	Resistor, 1K OHM, ¼W 5% Resistor/Brown, Black Red/
6	30-5470362	Resistor, 47K ¼W 5% Resistor/yellow, violet/orange
1	30-6270362	Resistor, 270K ¼W 5%
2	32-2000110	Capacitor, .001 UF Disk Ceramic
1	32-2001010	Capacitor, .01 UF Disk Ceramic
17	32-2010010	Capacitor, .1 UF Disk Ceramic
1	32-2233070	Capacitor, 33 UF .25V Tantalum
1	35-1000006	1N914 Zener Diode
44	35-3000001	LED Red
1	36-0089701	8T97
2	36-0740001	7400
2	36-0740201	7402
1	36-0740401	7404
2	36-0740402	74LS04
5	36-0740501	7405
1	36-0741001	7410
1	36-0741002	74LS10
1	36-0742701	7427
1	36-0743001	7430
1	36-0743002	74LS30
1	36-0780501	7805
1	36-0821201	8212
3	36-7410701	74107
3	36-7412301	74123
1	91-0400001	Cable Assembly "K"
1	92-0000002	P.C. Board, CP-A
1	93-3010006	Plastic Panel, Clear
1	93-3010007	Plastic Panel, Red
1	93-3010011	Bracket, Switch
1	93-3010015	Photo Mask
1	93-3010016	Paper Backing

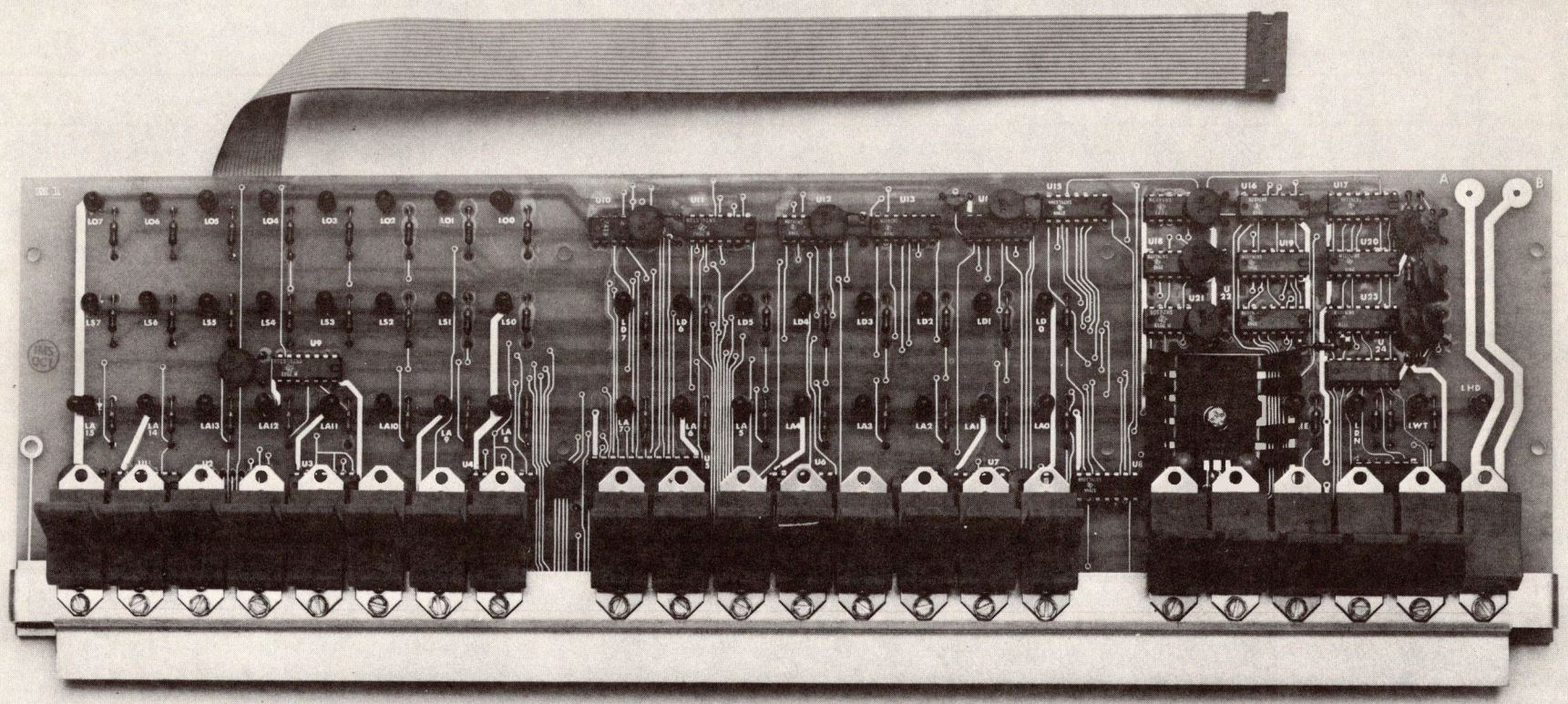
This is a revised parts list and will differ some respects from your manual. Please use this list for parts check off.

8/11/76

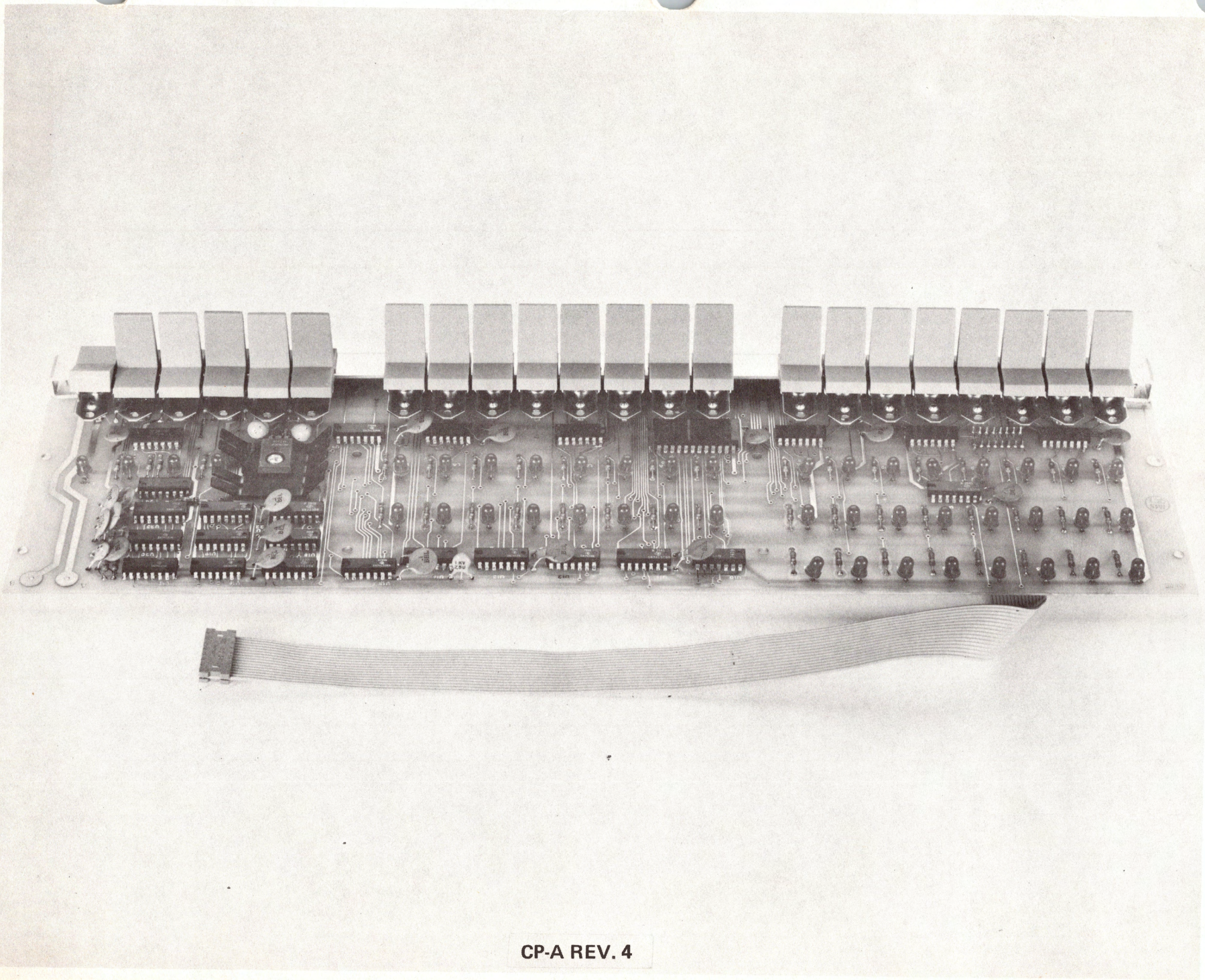
BOARD: CP-A

ITEM	IMSAI PART #	QUANTITY	DESCRIPTION/IDENTIFYING MARKS
Solder	15-0000001	10'	
Heat Sink	16-0100002	1	Thermalloy/6106B-14
Screw	20-2203001	22	4x¼", sheet metal, self-tapping, slotted he head integral washer
Screw	20-3203001	2	#6x¼" Self Tapping Sheet Metal Screws
Screw	20-3302001	1	6-32x5/16" Phillips Pan Head Machine
Screw	20-3916002	8	6-32x1¼" Button Head Allen Machine, Black
Nut	21-3120001	1	6-32 Hex
Lockwasher	21-3350001	1	Internal Star Lockwasher
Spacer	21-3600001	8	White Nylon ¼"
Spacer	21-3600002	8	White Nylon 7/16"
Switch	26-1500001	8	Blue Paddle Switch, on/none/on
Switch	26-1500002	8	Red Paddle Switch, on/none/on
Switch	26-1500003	2	Red Paddle Switch, momentary
Switch	26-1500004	3	Blue Paddle Switch, momentary
Switch	26-1600001	1	Red Rocker Switch, on/none/on
Resistor	30-3220362	44	220 Ohm, ¼Watt Resistor/red, red, brown
Resistor	30-4100362	9	1K Ohm, ¼Watt Resistor/brown, black, red
Resistor	30-5470362	6	47K Ohm, ¼ Watt Resistor/yellow, violet
Resistor	30-6270362	1	270K Ohm, ¼Watt Resistor/red, violet, yello
Capacitor	32-2000110	2	.001uf Disk Ceramic Capacitor
Capacitor	32-2001010	1	.01uf Disk Ceramic Capacitor
Capacitor	32-2010010	17	.1uf Disk Ceramic Capacitor
Capacitor	32-2233070	2	33UF Tantalum Capacitor
1N914	35-1000006	1	Signal Diode/1N914
LED	35-3000001	44	Light Emitting Diode/Red

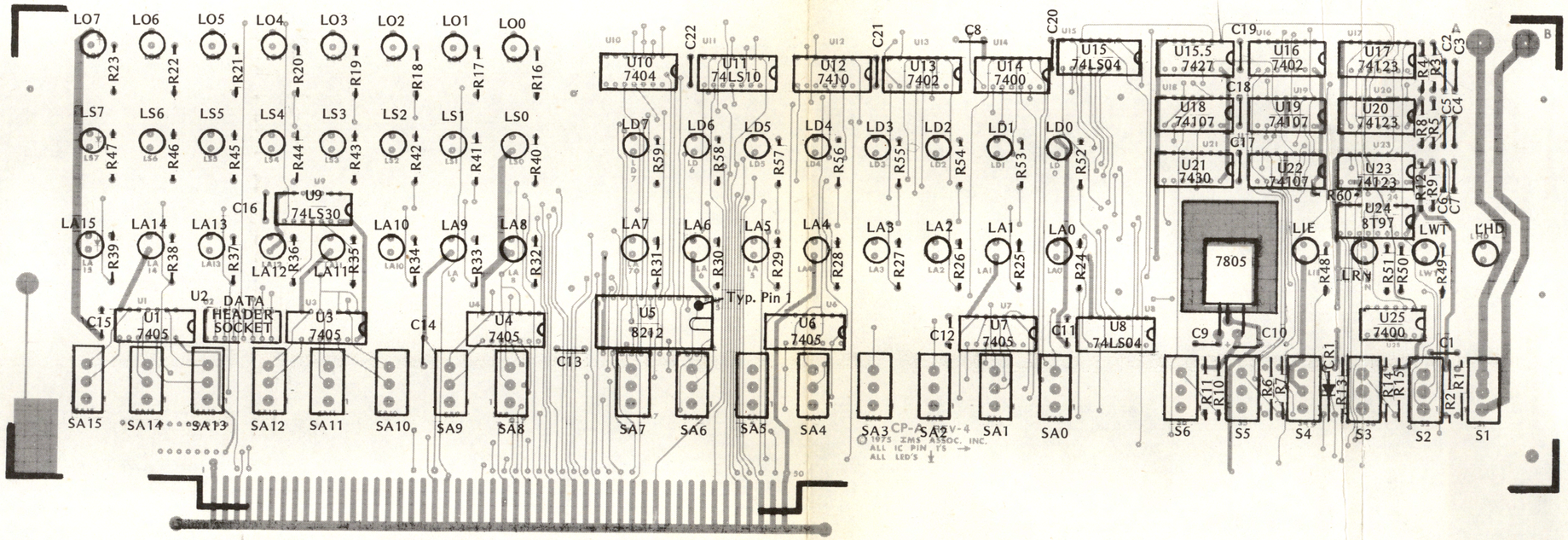
ITEM	IMSAI PART #	QUANTITY	DESCRIPTION/IDENTIFYING MARKS
8T97	36-0089701	1	Hex Tri-State Buffer/N8T97B
7400	36-0740001	2	Quad 2 Input NAND/SN7400N
7402	36-0740201	2	Quad 2 Input NOR/SN7402N
7404	36-0740401	1	Hex Inverter/SN7404N
74LS04	36-0740402	2	Hex Inverter (Low Power Schottky)/ SN74LS04N
7405	36-0740501	5	Hex Inverter Open Collector/SN7405N
7410	36-0741001	1	Triple 3 Input NAND/SN7410N
74LS10	36-0741002	1	Triple 3 Input NAND (Low Power Schottky)/ SN74LS10N
7427	36-0742701	1	Triple 3 Input NOR/SN7427N
7430	36-0743001	1	8 Input NAND/SN7430N
74LS30	36-0743002	1	8 Input NAND (Low Power Schottky)/SN74LS[illegible]
7805	36-0780501	1	5V Positive Volt Regulator/MC7805CP
8212	36-0821801	1	I/O Port/P8212/S1002
74107	36-7410701	3	Dual J-K Flip/Flop with Clear/SN74107N
74123	36-7412301	3	Dual Monostable Multivibrator, Retriggerable with Clear/SN74123N
Cable Assembly	91-0400001	1	Cable Assembly "K"
PC Board	92-0000002	1	Printed Circuit Board
Plastic Panel	93-3010006	1	Clear Plastic Panel
Plastic Panel	93-3010007	1	Red Plastic Panel
Bracket	93-3010011	1	Switch Bracket
Photo Mask	93-3010015	1	Photo Mask
Paper Backing	93-3010016	1	Paper Backing


LO7
LO6
LO5
LO4
LO3
LO2
LO1
LO0
LS7
LS6
LS5
LS4
LS3
LS2
LS1
LS0
LA13
LA12
LA11
LA10
LA3
LA2
LA1
LA0
LD7
LD6
LD5
LD4
LD3
LD2
LD1
LD0
LWT
LHD
A
B

CP-A REV. 4



CP-A REV. 4



IMS ASSOCIATES INC.

ASSEMBLY DIAGRAM

CPA REV 4 2/76

2/27/76

Gordon
Ex61

CP-A Rev. 4

ASSEMBLY INSTRUCTIONS

1) Unpack your board and check all parts against the parts lists enclosed in the package.

2) If gold contacts on the edge connector appear to be corroded, use pencil eraser to remove any oxidation. NOTE: Do not use Scotchbright or any abrasive material as it will remove the gold plating.

LED INSTALLATION

3) For a professional appearing finished CP-A Board two items in the assembly are important: first, the mounting of the LED indicator lamps, and second, the mounting of the paddle switches. Care is necessary in the mounting of both of these items to insure evenly spaced, straight line rows of components. If they are assembled carefully, the panel will have a professional appearance second to none. If these two items are assembled haphazardly, the panel will function; however, it will have a distinctly less than workmanlike appearance.

For maximum ease in uniform assembling, the LED indicator lamps should be installed on the board first, before any other components have been installed. They should not be pushed fully against the board, but, rather, should be set up approximately 1/8 inch to place them closer behind the acrylic panel mask; this provides for a greater viewing angle during panel operation. A small easy-to-make jig is extremely useful in accurate positioning of the LED indicators. This mounting aid consists of 1/8 inch thick material. A piece of 1/8 inch plastic, aluminum or masonite, or two pieces of 1/16 inch material such as vector board or old printed circuit board make ideal jigs.

A 3/4 square inch piece of the 1/8 inch material, or two of the 1/16 inch material should be cut and a narrow slot, such as would be produced by a hacksaw or coping saw blade, cut into one side a little bit past the center. As each light-emitting diode is installed in the board, leads can be inserted through the short slot cut into this piece and then through the board and the LED should be pushed up hard against the 1/8 inch piece so that its base sits flat and it will be held accurately 1/8 inch away from the surface of the front panel board. The lead should be soldered from the back while someone is holding the LED against the mounting aid from the front. The mounting aid can then be slipped out from under the LED.

Take care that every LED is mounted in the correct direction with the cathode down towards the 100 pin edge connector at

the bottom of the board. The cathode can be recognized by its proximity to the flat side on the base of the light emitting diode.

4) Insert and solder each of the forty-four red LED's at locations:

LO0 through LO7

LA0 through LA15

LD0 through LD7

LS0 through LS7

LIE, LHD, LRN, LWT

RESISTOR INSTALLATION

5) Insert and solder each of the forty-four 220 ohm ¼ watt resistors (red/red/brown) R16 through R59. See Assembly Diagram for location.

6) Insert and solder each of the six 47K ohm ¼ watt resistors (yellow/violet/orange) R3, R4, R5, R8, R9, and R12. See Assembly Diagram for location.

7) Insert and solder one 270K ohm ¼ watt resistors (red/violet/yellow) R1. See Assembly Diagram for location.

8) Insert and solder each of the nine 1K ohm ¼ watt resistors (brown/black/red) R2, R6, R7, R10, R11, R13 through R15, and R60. See Assembly Diagram for location.

IC INSTALLATION

9) Insert and solder each of the two 7400 IC's at locations U14 and U25.

10) Insert and solder each of the two 7402 IC's at locations U13 and U16.

11) Insert and solder the one 7404 IC at location U10.

12) Insert and solder each of the two 74LS04 IC's at locations U8 and U15.

13) Insert and solder each of the five 7405 IC's at locations U1, U3, U4, U6, and U7.

14) Insert and solder the one 7410 IC at location U12.

15) Insert and solder the one 74LS10 IC at location U11.

16) Insert and solder the one 7427 IC at location U15.5.

17) Insert and solder the one 7430 IC at location U21.

18) Insert and solder one 74LS30 IC at location U9.

19) Insert and solder each of the three 74107 IC's at locations U18, U19, and U22.

20) Insert and solder each of the three 74123 IC's at locations U17, U20, and U23.

21) Insert and solder the one 8T97 IC at location U24.

22) Insert and solder the one 8212 IC at location U5.

DISCRETE COMPONENT INSTALLATION

NOTE: Lead allowance on all capacitors must be long enough to allow them to be flattened (or laid down) flush on the board or chip to facilitate front panel mounting.

23) Insert and solder each of the seventeen .luf capacitors at locations C1, C2, C5, C6, C7, and C11 through C22. C16? OK

C2, C3, C5, C6, and C7 should be laid down.

24) Insert and solder each of the two .001 uf capacitors at locations C3 and C8.

25) Insert and solder the one .01 uf capacitor at location C4.

26) Insert and solder the two 33 mf 25 volt tantalum capacitors at locations C9 and C10. NOTE: Observe polarity as marked on board.

27) Insert and solder the 1N914 diode at position CR1.

REGULATOR AND HEAT SINK

28) Before installing heat sink bend all the heat sink fins horizontally (outward) to facilitate front panel mounting. The middle fin located on the right hand side of the board (when mounted) should be broken off or bent inward in order to allow space for the INTERRUPT/ENABLE LED (LIE) to be seen through the front panel.

29) To install the regulator and heat sink first bend the 7805 regulator leads at 90 degree angles to a length which allows their insertion into the hole pattern of the CP-A board. Then place heat sink as shown in Assembly Diagram and insert regulator as described above. Use a #6 screw on the component side of the board and lockwasher and nut on the solder side of the board. Tighten the screw carefully to insure proper alignment of the heat sink to prevent shorting to adjacent traces.

CP-A TO MPU-A INSTALLATION

30) Using the 16 conductor ribbon cable with 16 pin 3M dual inline connector, insert one end into the hole pattern U2 from the back side of the CP-A board so that it can be soldered from the front (component side) of the CP-A board. The cable should be mounted so that it extends upward from the top of the chassis when the board is mounted.

SWITCH INSTALLATION

NOTE: There are three types of switches included for installation on the front panel (disregarding color). They include:

A. One 2 position red rocker switch. This is the AC power switch.

B. 5 momentary 3 position with spring return to center paddle switches - identified by the lack of a Nipple (raised portion) on the front of the switch mounting tab.

C. 16 2 position no spring return paddle switches.

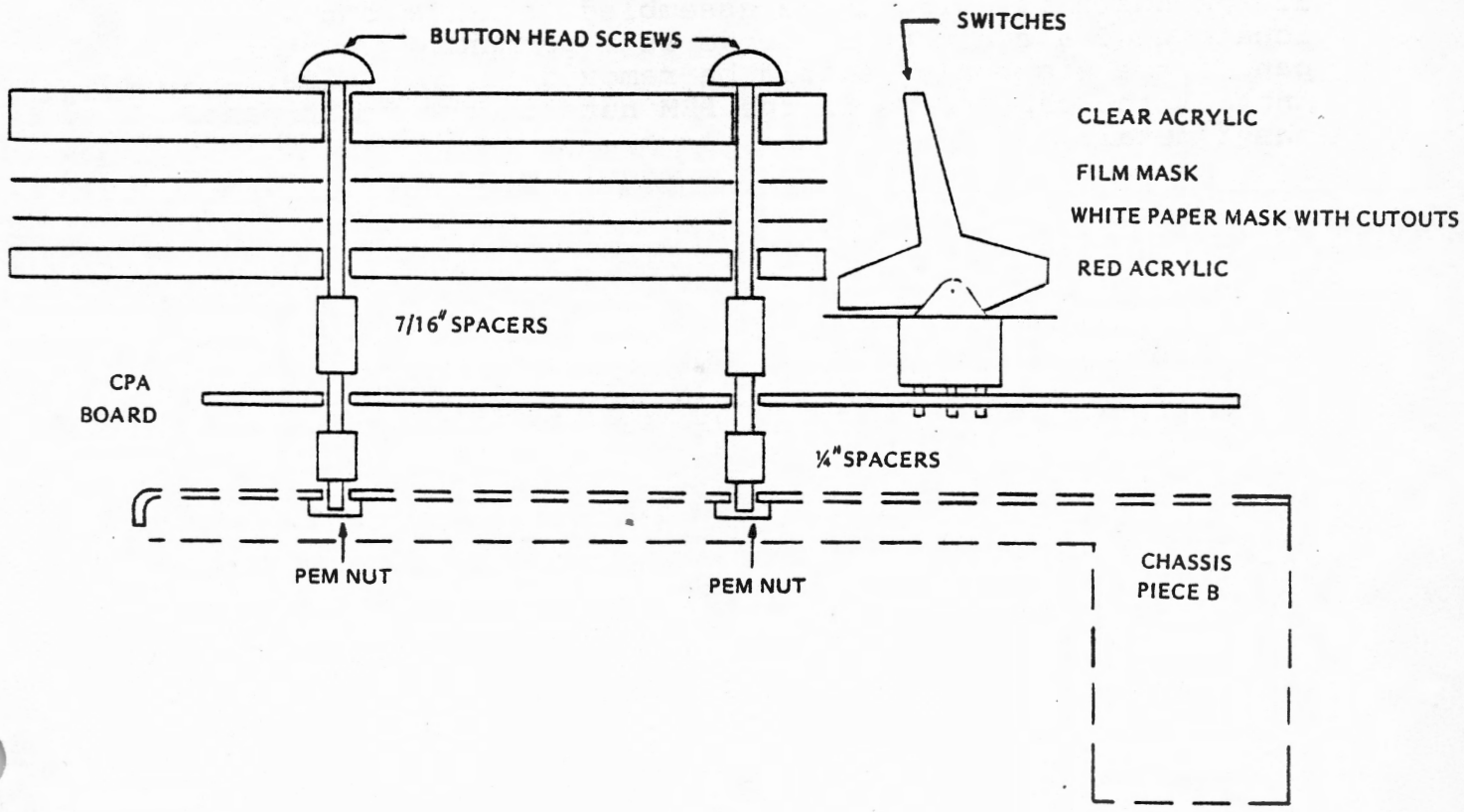
NOTE: Temporarily plug a 100 pin edge connector on the CP-A Board while switches are being soldered to help insure proper spacing between the PC Board and switch bracket.

31) The last step is the assembly of the switches and the switch mounting bracket. Note that the front panel includes switches whether mounted in the front of the cabinet or not. The POWER/ON/OFF Rocker Switch mounts at the extreme right switch position. The Paddle Switches are provided in both two-position and center-off spring return types. The two-position switches are used for the ADDRESS-DATA and ADDRESS PROGRAMMED INPUT location while the center off-spring return are used for the Control Functions.

32) When the entire row has been spaced accurately, the board should be turned over and a center switch should be soldered in place taking care that the board is not bowed towards or away from the switches. When the board is positioned correctly, there will be a small space approximately 3/64 inch or slightly under 1/16 inch between the bottom of the switch and the front of the front panel board. The two end switches should be similarly checked to make sure that the spacing to the board is correct and soldered in place, and then one switch each at the ¼ positions checked as to spacing from the board and soldered into place. Then the remainder of the switches can be soldered. Examine visually for solder splash or bent/unsoldered pins.

PANEL ASSEMBLY

Refer to the diagram to see how the clear front acrylic piece, the photograph mask, the die cut paper backup and the red acrylic panel are assembled in sequence with the 6/32 x 1½ inch button head screws.



Both the Photographic mask and the paper backup sheet should be trimmed to size after assembly. Marks are provided on both, and they should be cut out carefully using a straight edge and a very sharp knife against a wooden cutting board. Scissors may be used if a guide line is first drawn on the sheets. The 8 holes for the assembly screws should be cut out on the mask and the paper sheet as indicated in the diagram. Then the protective paper may be removed from the two acrylic sheets and the sandwich assembled carefully. Avoid getting dush caught in between any of the pieces. A soft lint-free rag very slightly moistened can be an aid in cleaning any dust off plastic or file surfaces.

When the acrylic pieces, film and paper have been assembled, eight 7/16 inch spacers may be slipped over the screws and then the whole assembly inserted through the mounting holes on the CP-A board. Take care that there is no interference from any component standing too high and that the acrylic panels sit down completely on the 7/16 inch spacers against the board.

Eight ¼ inch spacers can then be slipped over the screws behind the CP-A board and eight #6 nuts and lockwashers can be put on to hold the sandwich together.

The panel board should now be ready to plug in and use. If the board is going to be assembled in the front location of a cabinet, to serve as a permanent front panel, the eight nuts should be removed at this time. Install the cap screws in the PEM nuts in the front panel sheet metal.

USER GUIDE

The CP-A board contains no user option jumpers or any other special connections that must be made to use the board. If the panel is mounted in the IMSAI 8080 cabinet then the power on/off switch should be connected using a separate wire to the power supply section as described in the Power Supply documentation. If the panel board is not going to be mounted in the cabinet, then the power switch should not be connected. In this case, the power switch on the inside or on the back of the cabinet would be used for controlling power to the IMSAI 8080.

Panel installation requires a backplane slot. The panel may be plugged directly into the front slot of the mother board (with the mounting screws from the acrylic face plate assembly extending through the metal panel immediately behind and secured with lock washers and nuts) or plugged on an extender card into any location in the back plane.* The 16 pin DIP plug on the end of the flat cable must also be inserted into the corresponding socket in the upper right hand corner of the MPU-A board.

Only one front panel should be plugged into the bus at any time to avoid conflicts between multiple driving sources on the same signal lines for some of the control lines between the front panel and the system bus. The front panel is now ready to operate.

The 16 ADDRESS-DATA and ADDRESS-PROGRAMMER INPUT switches are 2 position paddle switches and represent a 0 in the down position and a 1 in the up position. The switches are provided in two colors, and can be arranged either in color groups of four to assist programming in hexadecimal or color groups of 3, 3 and 2, to aid in octal programming.

The low order byte of address switches, serve to enter into memory either data or program instructions. These switches are labeled ADDRESS-DATA 0 through 7. Each byte of data or program that is to be entered from the front panel is set into these switches after the appropriate address has been selected and entered. The switch

---

*Switches should be included whether the front panel is mounted in the front of the cabinet or not.

positions are not indicated on the indicator lights until the information is deposited in memory. At that time the information from these switches appears on the data bus. The high order byte of address switches is labeled ADDRESS-PROGRAMMED INPUT and these switches can be read by the program as input port position hex FF or octal 377. The additional labels 0 through 7 are provided above these switches to assist in interpreting the switch positions when being used as an input port. The position of these switches does not appear in the indicators until the input instruction from position FF is executed, during which execution time the switch positions appear on the data bus as it is being read into the 8080 processor.

The six control switches are grouped at the right end of the panel. They are center-off two-position spring-return switches with the exception of the POWER ON/OFF switch, which is a rocker type to eliminate accidental powerdowns. The function switches are provided in alternating colors for easy identification and to reduce operator error.

The RUN/STOP switch controls program execution. When the switch is pushed to the RUN position, a control signal is sent to the processor board and enables it to start or continue executing program instruction beginning in the location indicated at that time in the address bus lights. When the address switch is depressed to the STOP position, this enable signal is removed from the processor board at the beginning of the next instruction cycle so that the processor will stop executing during the fetch cycle for that following instruction.

When the processor is enabled to RUN, the RUN light above the RUN/STOP switch will be lit. When the processor has been stopped, the WAIT light to the right of the RUN light will be lit. During normal operation, the RUN light will be on full and the WAIT light will be on partially, the exact amount depending on how many wait cycles are required by the memory and peripheral devices being run by the processor at the moment.

The front panel must be holding the processor in the stopped condition for the SINGLE STEP switch, the DEPOSIT/DEPOSIT NEXT switch, or the EXAMINE/EXAMINE NEXT switch to operate.

The EXAMINE/EXAMINE NEXT switch provides the facility for observing what is stored in memory in any location or for setting the program counter to any desired location to initialize program execution there.

When examining the contents of a location in memory, the 16 address switches are used to enter the 16 bit address. This 16 bit address is normally said to be divided into two 8-bit sections labeled high order and low order. The high order address is on the left hand side of the panel, and the low order address is in the center. The low order byte contains bits 0 through 7 and the high order byte contains bits 8 through 15. When only a small amount of memory is being used the high order bits are normally 0 and the switches must be in down position, unless the address jumper selection on the memory board is wired otherwise.

When the EXAMINE switch is actuated, the processor jumps to the address location set in the 16 address switches and is stopped during the fetch cycle out of that memory location. At that time, the address bus indicators will show the address set in the 16 address switches and the data bus indicators will show the contents of that memory location. Any additional locations in memory may be observed by setting the 16 address switches to that desired address and actuating the EXAMINE switch again. When the EXAMINE NEXT switch is actuated, the address shown in the address bus indicators is incremented by 1 and the contents of that following memory location are displayed on the data bus lights. Thus, a program or data would normally be checked by setting the first address in the address switches and actuating the EXAMINE switch to see the first byte, and thereafter actuating the EXAMINE NEXT position to observe each succeeding byte of data or program.

The DEPOSIT/DEPOSIT NEXT switch is similar in its operation but provides for changing the data or program stored in the memory. When the switch is actuated to the DEPOSIT position, the values of the lower address byte switches, that is, bits 0 through 7 labeled Address-Data, are deposited into the address currently being indicated on the 16 address bus indicators. After the DEPOSIT switch is actuated, the data will appear on the data bus indicators. If the data was incorrect because the switches were set wrong, the switches can be changed, the DEPOSIT switch actuated again, and the new values will be deposited to memory in that same location.

When the DEPOSIT NEXT position is actuated, the address currently appearing in the 16 address bus indicators is first incremented by 1 and the data entered into the ADDRESS/DATA switches is deposited in that following location and will appear in the data bus. The DEPOSIT NEXT position functions exactly the same as depressing EXAMINE NEXT to increment the address bus by 1 and then actuating DEPOSIT to deposit the ADDRESS-DATA switch positions into that location.

When the processor is stopped, instructions may be executed one at a time through the use of the SINGLE STEP switch to the right of the RUN/STOP switch. If this switch is depressed or raised, the processor board is permitted to run one instruction, and it will stop when it is in the fetch cycle in the following instruction. Thus repeated operations of this switch permit the programmer to step through his program one instruction at a time and follow what the machine is doing, noticing on the data bus what the fetched instruction is, and on the address bus the location from which that instruction is being fetched. For instructions requiring multiple memory accesses, for instance those with an address following in the second or third byte, each operation of the SINGLE STEP switch advances through only one part of the instruction. Thus, each byte of the instruction being read in and each byte of data being read in or out may be observed on the panel.

The RESET/EXTERNAL CLEAR switch provides the system reset functions. When depressed to the EXTERNAL CLEAR position the CLEAR signal is given to all external input/output interface cards which are wired to be reset by this signal. When raised to the RESET position, the 8080 processor is reset. This sets the program counter to location 0 and then returns control to the processor. If the front panel is permitting the processor to run when the RESET switch is actuated, upon release of the RESET switch the processor continues execution starting at position 0. If the front panel was holding the processor in a stopped state, during the time the RESET switch was actuated, then the program counter will be set to 0. When the RESET switch is released, the processor will remain stopped and will be positioned at memory location 0.

The 8-BIT PROGRAMMED OUTPUT INDICATOR lights can be controlled by the program through the use of the output instruction to port location hex FF or octal 377. When 0 bits are output into this port, the indicator lights will be turned on and when 1 bits are output into this port, indicator lights will be turned off.

The STATUS BYTE INDICATOR LIGHTS display the condition of the status byte during the execution of that instruction. The 8 status bits included in the status byte are the Memory Read Bit, the Input Instruction Bit, the Instruction Fetch Bit M1, the Output Instruction Bit, the Halt/Acknowledge Bit, the Stack Operation Bit, the Write Output Complement Bit, and the Interrupt Acknowledge Bit. In normal front panel operation, whenever the machine is stopped and the EXAMINE, DEPOSIT, EXAMINE NEXT or DEPOSIT NEXT switches are being used, the MEMORY READ, the M1 INSTRUCTION FETCH, and the WRITE OUT COMPLEMENT STATUS lights should be on.

While single-stepping through a program, either these or other status lights will be on as appropriate to the instruction function being executed at that moment.

For a more complete description of the functions of the status bits, reference should be made to the Intel 8080 Micro Computer Systems User's Manual. The INTERRUPT ENABLED indicator is turned on whenever the interrupts are enabled into the 8080 processor by the INTERRUPT ENABLE INSTRUCTION. This light is turned off either by an interrupt occurring and the processor acknowledging it, or by the instruction to disable interrupts. The HOLD indicator light is lit whenever a special-purpose input/output card is holding the processor so as to gain direct access to the memory on the system bus.

CPU-Z Card

GODBOUT

COMPUPRO'S CPU-Z PROCESSOR CARD

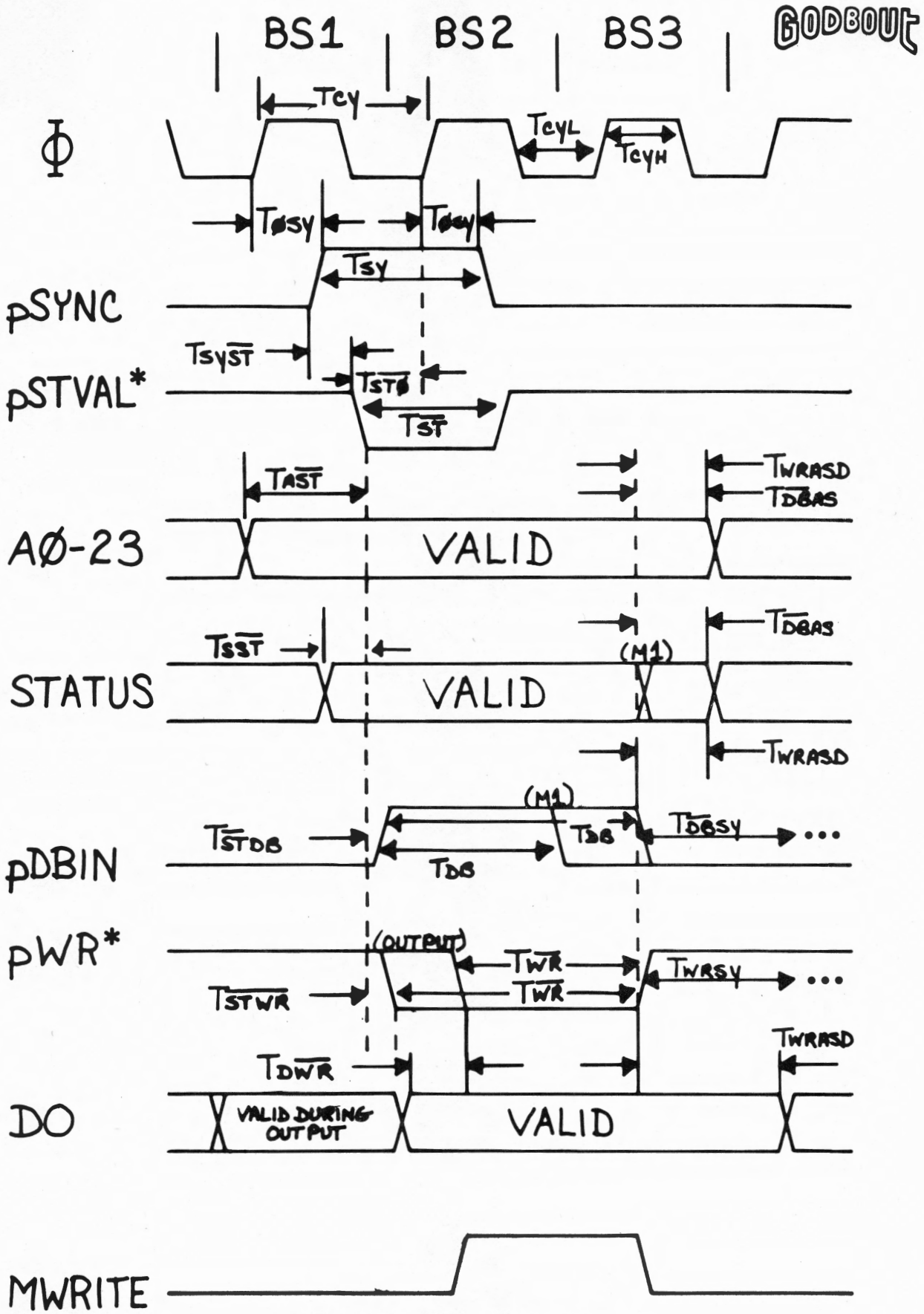
The **CPU-Z** was specifically designed to give the user the finest Z80 CPU board available for the S-100 bus. The board features on-board sockets for 4K of EPROM or RAM, a Memory Manager to extend the addressing range of the system from 64K-bytes to a maximum of 16M-bytes of system RAM, fully maskable vectored interrupts to speed up system throughput by eliminating polling loops, and wait state generation circuitry for all machine cycles (M1, I/O, & MREQ), and on-board memory.

Power-on-jump circuitry is included to allow the CPU to begin execution at any 256 byte boundary within the lower 64K. A switch is provided to disable this feature. A switch is also provided to allow the power-on-jump circuitry to be active at power-on only, or to work at power-on and with each RESET (jump-on-reset).

The Extended Addressing bits (A16-A23) can be altered with an output to port FD(H) and the Interrupt Vector Mask can be set by an output to port FE(H).

The **CPU-Z** can be run at any speed from 2MHz to 6MHz with the standard A&T board running at 4MHz and switch selectable to run at 2MHz.

The **CPU-Z** was specifically designed for electrical and mechanical compatibility with the IEEE S-100 bus standard as a permanent master, but should also work with most pre-IEEE hardware. Provision is made to accommodate an IMSAI-type front panel. The S-100 bus is the professional level choice for commercial, industrial, and scientific applications. This bus provides for ready expansion and modification as the state of the computing art improves. We believe this board, along with the rest of the S-100 portion of the **CompuPro** family, is one of the best boards available for that bus.



CPU-Z Timing
Tcy = 250ns

© 1980

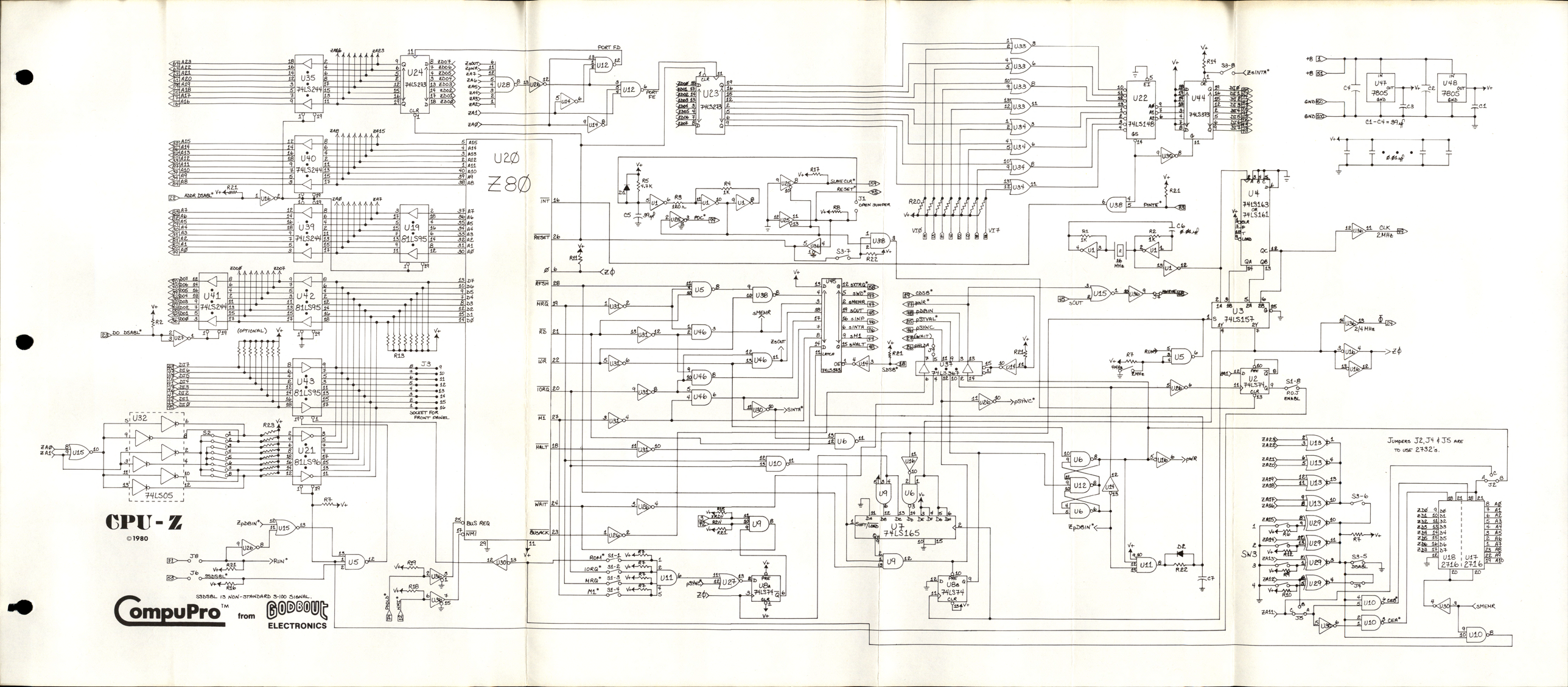
GODBOUT ELECTRONICS BOX 2355, OAKLAND AIRPORT, CA 94614

TIMING PARAMETERS AS SPECIFIED FOR THE S-100 IEEE STANDARD
(ALL PERIODS OF TIME IN NANOSECONDS)

		I.E.E.E		GODBOUT CPU - Z	
		MIN.	MAX.	2MHZ	4MHZ
tCY	clock cycle period	166	2000	500	250
tCYH	clock cycle high	0.4tCY	----	250	120
tCYL	clock cycle low	0.4tCY	----	250	130
tØSY	delay Ø high to pSYNC high	10	0.4tCY	80	70
	delay Ø high to pSYNC low	10	0.4tCY	120	80
tSY	pSYNC high	0.7tCY	----	550	250
tSTØ	pSTVAL* low to Ø high	0	----	340	140
tST	pSTVAL* low	50	----	500	250
tAST	address stable to pSTVAL* low	70	----	300	200
tSST	status stable to pSTVAL* low	40	----	300	150
tDB	pDBIN high	0.9tCY	----	900	400
	(M1 cycle)	0.9tCY	----	650	275
tSTDB	pSTVAL* low to pDBIN high	20	----	50	50
tDBAS	hold time for:				
	address	50	----	200	100
	status	50	----	100	100
	from pDBIN low				
tWR	pWR* low	0.9tCY	----	500	250
tSTWR	delay pSTVAL* low to pWR* low	30	----	440	200
	(during output cycle)			40	40
tDWR	DO valid to pWR* low	0.1tCY	----	420	180
	(during output cycle)			500	300
tWRASD	hold time for:				
	address	0.2tCY	----	250	130
	status	0.2tCY	----	100	100
	data out	0.2tCY	----	470	280
	from pWR* high				
tWRSY	pWR* high before next pSYNC	0.0	----	380	200
tDBSY	pDBIN low before next pSYNC	0.0	----	380	200
tACC	pSTVAL* low to data valid			205	---
tSYST	pSYNC high to pSTVAL* low	20	----	50	---
tWRMR	pWR low/high to MWRITE low/high	30	----	10	---

© 1980

GODBOUT ELECTRONICS BOX : AKLAND AIRPORT, CA 94614


CPU - Z
©1980
SSDSBL IS NON-STANDARD S-100 SIGNAL.
CompuPro™ from GODBOUT ELECTRONICS
U2Ø
Z8Ø
PORT FD
PORT FE
ADDR DSABL*
DO DSABL*
(OPTIONAL)
SOCKET FOR FRONT PANEL
JUMPERS J2, J4 & J5 ARE TO USE 2732's.
C1-C4 = 39 µf
.01 µf
BUS REQ
NMI
RUN*
SSDSBL*
ZpDBIN*
SLAVE CLR*
RESET*
OPEN JUMPER
ZØ
SINTA*
pSYNC*
sMEMR
CLK 2MHz
2/4 MHz
PWR
ROJ ENABL
SW3
SOUT
PINTE*

MPU

MPU-A

FUNCTIONAL DESCRIPTION

The MPU-A board is the processor board for the IMSAI 8080 Microcomputer System. It is designed using the Intel 8080 micro-processor chip. The bus arrangement and board connector has been chosen to be 100% compatible with the MITS Altair M8800 Microcomputer system so that all boards are 100% interchangeable between the Altair system and the IMSAI 8080 system.

Every effort has been made to keep the design simple and straight-forward to maximize reliability and ease of maintenance. MSI and LSI are used where appropriate, and discrete components are held to a minimum for greater circuit reliability and ease of assembly.

The 8224 clock driver chip and an 18 Megahertz cyrstal are used to generate the 2-phase, 2 Megahertz non-overlapping clock for the 8080A. An 8212 is used as a latch for the status signals and two 8216 tri-state bi-directional bus drivers are used to interface the 8080A with the IMSAI 8080 input and output data buses. All other address, status, and control lines are driven by tri-state bus drivers.

Unregulated +16, -16, +8 volts, and ground must be supplied to the bus. On-board regulation is used to arrive at the power supply levels needed to run the chips. Integrated circuit power regulators with overload protection are used. The board is supplied with ample bypass filtering using both disc ceramic and tantalum capacitors.

The board connector is a 100 pin edge connector on .125 inch centers 50 pins on each side. Dimensions are 5 inches by 10 inches, using 2 sided glass reinforced epoxy laminate, with plated feed through-holes to eliminate the need for any circuit jumpers. The contact fingers are gold-plated over nickel for reliable contact and long life. All other circuitry is tin-lead plated for better appearance and more reliable solder connections.

Power-on reset is included on this board along with pull up resistors for all inputs required so that with the front panel removed from the IMSAI 8080 machine, the power-on reset will start the program at position 0 out of a ROM. All other necessary conditions are met so that the system will run without the front panel attached, for use in dedicated controller applications where no operator-processor interaction is desired.

THEORY OF OPERATION

The IMSAI MPU-A board is structured around the Intel 8080A microprocessor chip, and much of the MPU-A board is wired to support the 8080A device. The MPU-A board provides interfacing between the 8080A chip and the data and address busses, clock and synchronization signals, and the voltage regulation necessary for the 8080A and other chips. The internal functioning of the 8080A is thoroughly described in the Intel 8080 Microcomputer System User's Manual. Reference should be made to this manual for information concerning the operation and use of the 8080A.

The address lines from the 8080A drive the address bus on the back plane through 8T97 tri-state buffer drivers. These drivers may be disabled through the ADDRESS DISABLE line on pin 22 of the back plane. Intel 8216 bi-directional bus drivers connect the 8080's bi-directional data bus to the back plane's dual uni-directional DATA IN and DATA OUT busses. The direction of data transmission is determined by the DIRECTION ENABLE line. The DIRECTION ENABLE line is in turn controlled by the front panel and the processor status signals DATA BUS IN and HALT ACKNOWLEDGE. The 8216 can be disabled by the DATA OUT DISABLE line on pin 23 of the back plane.

The 8080A's bi-directional data bus is also connected to the data bus socket and the 8212 status byte latch. The data bus socket is used to connect the front panel to the bi-directional bus, while the 8212 latch transfers the status byte to the back plane via 8T97 drivers. These drivers are disabled by the STATUS DISABLE line on pin 18 of the back plane. The 8212 is latched up by the STATUS STROBE signal of the 8224 clock chip to store the status information for each instruction cycle.

One K pullup resistors to +5 volts are connected to all the bi-directional bus lines to ensure that during the time the bus is not driven, the 8080A reads all 1's.

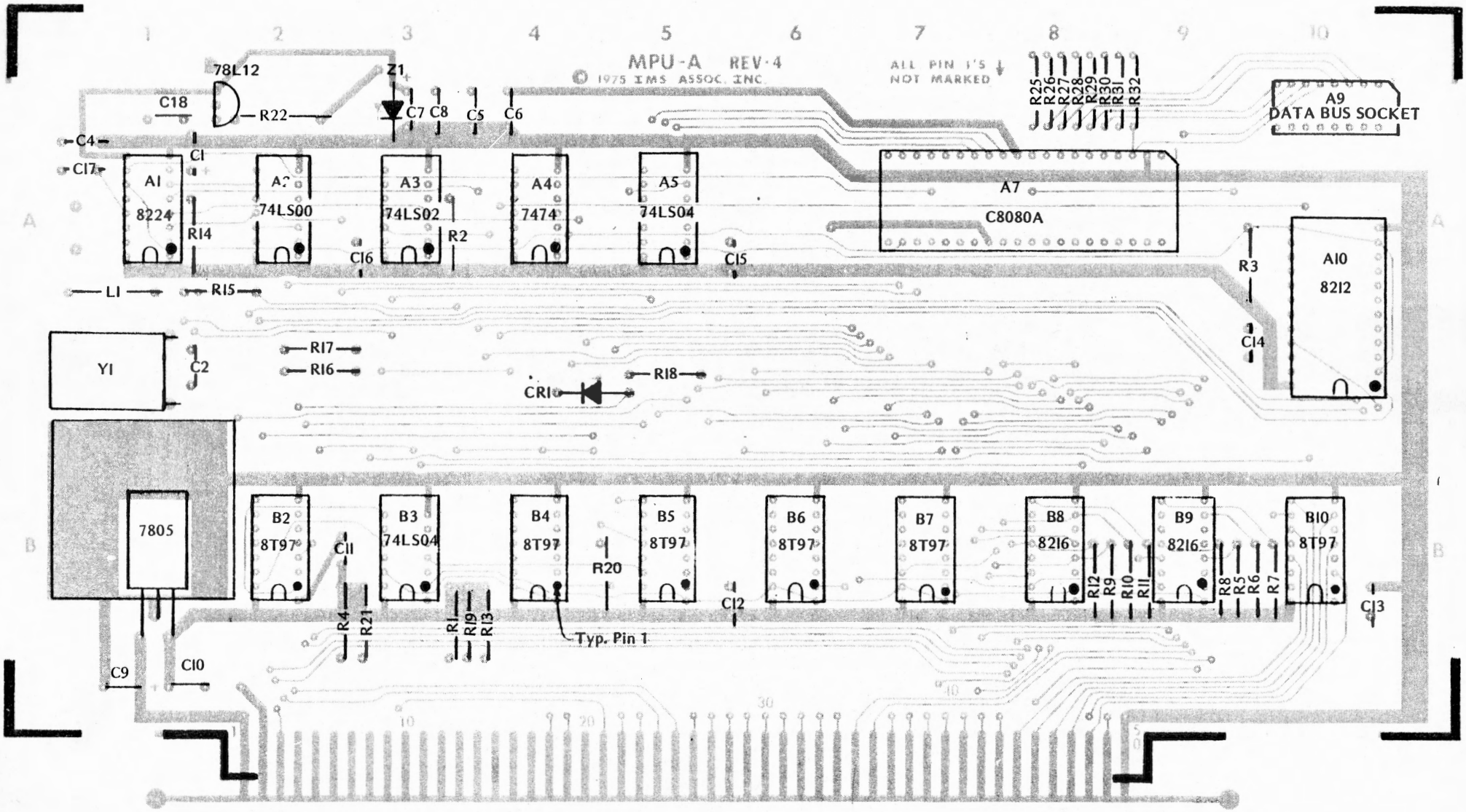
The 8224 clock chip and crystal oscillator, provide the two-phase non-overlapping 2 megacycle system clock for the 8080A. These clocks are also driven onto the back plane through 8T97 tri-state buffered drivers.

The CLOCK line on the back plane is driven from the TTL Phase II clock line through a delay so that the phase relation of the clock signal to the Phase II and Phase I back plane signals, is nearly identical to that produced by the MITS Altair 8800 system. Six sections of a 7404 are used for this delay to provide greater simplicity and higher reliability than a one-shot. The 8224 chip also provides the power-on reset function through use of a 4.7K resistor and 33 uf capacitor connected to the reset input of the 8224. The power-on reset is applied to the 8080A and is applied to the POWER ON CLEAR line, pin 99 on the back plane.

The two BACK PLANE READY signals are ANDed and connected to the 8224 for synchronization with the Phase II clock before being connected to the 8080A chip. The INTERRUPT line is connected directly to the 8080A, while the HOLD REQUEST line is synchronized with the Phase II clock and then connected to the 8080A.

The six processor status signals (SYNC WRITE, STROBE DATA BIT IN, INTERRUPT ENABLED, HOLD ACKNOWLEDGED, and WAIT ACKNOWLEDGE) are all driven onto the back plane through 8T97 tri-state buffered drivers. These drivers may be disabled by the CONTROL DISABLE line, pin 19 on the back plane.

The +5 volts is regulated from the +8 volts by a 7805 integrated circuit regulator, while the -5 volts is regulated by a 5 volt zener and a 470 ohm resistor from the 16 volt bus. The +12 volts is regulated by a 12 volt Zener and connected to the +16 volt line by two 82 ohm ½ watt resistors in parallel. All voltages are filtered with .33 microfarad tantalum and disc ceramic capacitors.

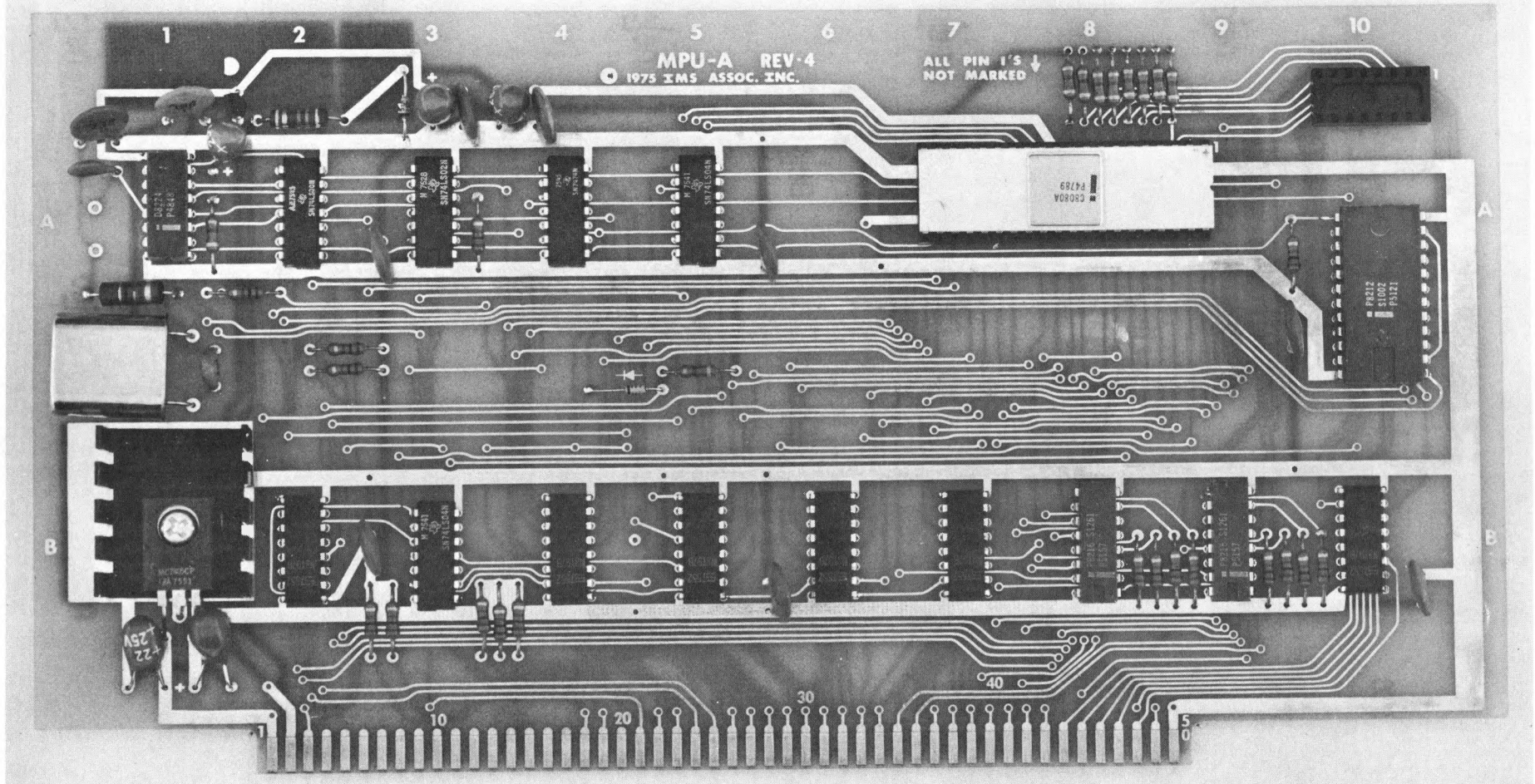


IMS Associates Inc.

ASSEMBLY DIAGRAM

MPU-A REV 4 2/76

5/18/76


MPU-A REV-4
© 1975 IMS ASSOC. INC.
ALL PIN 1'S
NOT MARKED
1
2
3
4
5
6
7
8
9
10
A
B
10
20
30
40
50

MPU-A REV 4

BOARD: MPU

ITEM	IMSAI PART #	QUANTITY	DESCRIPTION/IDENTIFYING MARKS
74ls00	36-0740001	1	Quad 2 Input NAND (Low Power Schottky) SN74LS00N
74ls02	36-0740202	1	Quad 2 Input NOR (Low Power Schottky) SN74LS02N
74ls04	36-0740402	2	Hex Inverter (Low Power Schokkty) SN74LS04N
7474	36-0747401	1	Dual D Flip Flop SN7474
7805	36-0780501	1	5V. Positive Volt Regulator MC7805CP
8080	36-0808001	X	Microprocessor C8080A
8212	36-0821201	1	Input/Output Port P88212/S1002
8216	36-0821601	2	Bi-directional Bus Driver D8216/S1002
8224	36-0822401	1	Clock Generator & Driver D8224
1n914	35-1000006	1	Silicon Diode 1N914 (blue/black)
8t97	36-0089701	6	Hex Tri-State Buffer N8T97B
dc-39p	32-0239010	1	Disk Capacitor 39M/1KV
dc-56p	32-0256010	10	Disk Capacitor .1
h-hsto-220	16-0100002	1	Heat Sink Thermalloy/610B-14
1-1uH	29-0400001	1	Inductor WEE/1.0/uH10%/7440/NYT
pc-mpu-a	92-0000011	1	Printed Circuit Board MPU
78L12	36-0781202	1	Transistor NS78L12A
r-470-½	30-3470462	1	470 Ohm, ½ Watt Resistor (yellow/violet brown)
r-1K¼	30-4100362	18	1K Ohm, ¼ Watt Resistor (brown/black/red)

ITEM	IMSAI PART #	QUANTITY	DESCRIPTION/IDENTIFYING MARKS
r-4.7K¼	30-4470362	10	4.7K Ohm, ¼ Watt Resistor (yellow/violet/red)
sts-16	23-0800001	1	Solder Tail Socket/16 Pin
sts-40	23-0800000	1	Solder Tail Socket/40 Pin
tc-35-25	32-2233070	5	Tantalum Capacitor 33-25
x-tal-18m	35-5000001	1	18 MHz Crystal 00W35/X-TRON
z-5.1-.4	35-1000005	1	Zener Diode 1N751
s/n/lw	20-3402001	1 ea.	3/8" screw/nut/lockwasher
Solder	15-0000001		

MPU-A ASSEMBLY INSTRUCTIONS

1) Unpack your board and check all parts against the parts lists enclosed in the package.

2) If gold contacts on the edge connector appear to be corroded, use pencil eraser to remove any oxidation. NOTE: Do not use Scotchbright or any abrasive material as it will remove the gold plating.

RESISTOR INSTALLATION

3) Insert and solder nineteen 1K ohm 1/4 watt resistors (brown/black/red) R1 through R13, R15 through R17, R19, R20 and R21. See Assembly Diagram for location.

4) Insert and solder ten 4.7K ohm 1/4 watt resistors (yellow/violet/red) R14, R18, and R25 through R32. See Assembly Diagram for location.

5) Insert and solder one 470 ohm 1/2 watt resistor (yellow/violet/brown) R22. See Assembly Diagram for location.

IC INSTALLATION

NOTE: All IC pin 1's point in the direction of the edge connector unless otherwise indicated on the board.

6) Insert and solder the one 74LS00 in location A2.

7) Insert and solder the one 74LS02 at location A3.

8) Insert and solder each of the two 74LS04's at location B3 and A5.

9) Insert and solder the one 7474 at location A4.

10) Insert and solder the one 8224 at location A1.

11) Insert and solder each of the six 8T97's at locations B2, B4, B5, B6, B7, and B10.

12) Insert and solder each of the two 8216's at locations B8 and B9.

13) Insert and solder the one 8212 at location A10.

DISCRETE COMPONENT INSTALLATION

14) Insert and solder the 16 pin IC socket at location A9.

15) Insert and solder the 40 pin IC socket at location A7. (Do not install 8080 at this time.)

16) Insert and solder the one 1N914 diode (CR1) as shown in the Assembly Diagram. NOTE: Observe polarity as indicated on the board.

17) Insert and solder the one 5.1 volt zener diode 1N751 (Z1) as shown on the Assembly Diagram. NOTE: Observe polarity as indicated on the board.

18) Insert and solder the one 18 MHz crystal (Y1) as shown on the Assembly Diagram.

19) Insert and solder the one 1uH inductor (L1) as shown on the Assembly Diagram.

20) Insert and solder the one 39 pf disk capacitor (C2) as shown on the Assembly Diagram.

21) Insert and solder the one 56 pf disk capacitor (C17) as shown on the Assembly Diagram.

22) Insert and solder each of the ten .1 uf disk capacitors at locations C4, C6, C8 and C11 through C16, and C18.

23) Insert and solder each of the five 33 uf tantalum capacitors C1, C5, C7, C9 and C10 as shown on the Assembly Diagram. NOTE: Observe polarity as marked on the board.

REGULATOR AND HEAT SINK INSTALLATION

24) Insert and solder the one 78L12 -12 volt regulator observing orientation as shown on the Assembly Diagram and on the board.

25) Bend the leads of the 7805 regulator at 90 degree angles approximately ¼" from the bottom edge of the regulator to facilitate insertion on top of the heat sink.

26) Insert the #6 screw through the regulator and heat sink and attach washer and nut from the back side of the board. NOTE: Be sure to hold the heat sink in proper vertical position while tightening the screw in order to prevent shorting to adjacent traces. Solder in the 7805 leads.

Before Installing the 8080 Chip

If possible, before plugging in the 8080A chip, the board should be inserted in a chassis, the power turned on, and the the voltage levels checked on the 40 pin socket. Pin 2 should be ground and pin 11 should be -5 volts. Pin 20 should be +5 volts and pin 28 should be +12 volts. If one of these three voltages is not correct, ascertain the cause and correct it before plugging in the 8080A chip. When these voltages measure correctly, the 8080A chip should be inserted carefully into the 40 pin socket (with the board removed and the power off!)

27) Finally insert the 8080A Microprocessor chip in the 40 pin IC socket located at A7. Orient pin 1 as indicated on the board.

NOTE: The 16 pin IC socket located at A9 is where the front panel data bus cable plugs into the MPU-A board.

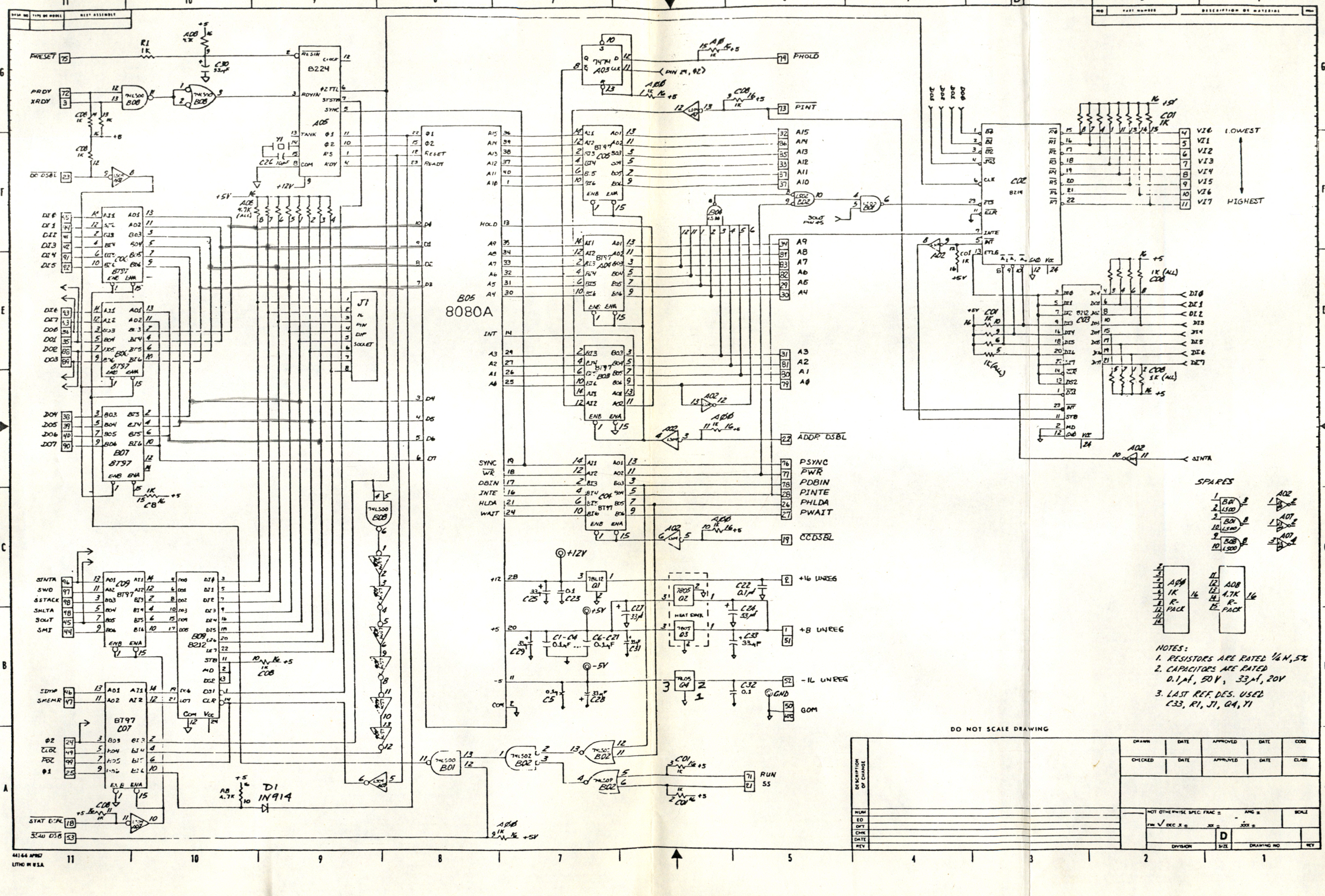
USER GUIDE

The IMSAI MPU-A board requires no jumpers or user options for its use. The board is ready to function after connection to the back plane and the bi-directional bus. The bi-directional bus lines are provided by a 16-conductor cable from the CPA board, connected via a 16-pin DIP plug in location A-10. Verify proper insertion of this plug (i.e., pin 1 to pin 1) before use of the board.

The clock crystal frequency is 18 megahertz, and the 8224 device derives from this 18 MHz signal the necessary 2 MHz two-phase non-overlapping system clock. These 2 MHz clocks are brought out onto the back plane for use by other system boards. The board must be used with an 8080A chip as the 8080 chip is not compatible with the 8224 clock generator. Information on the timing of the logic signals and the description of the 8080A instruction set can be found in the Intel 8080 Micro Computer Systems User's Manual.

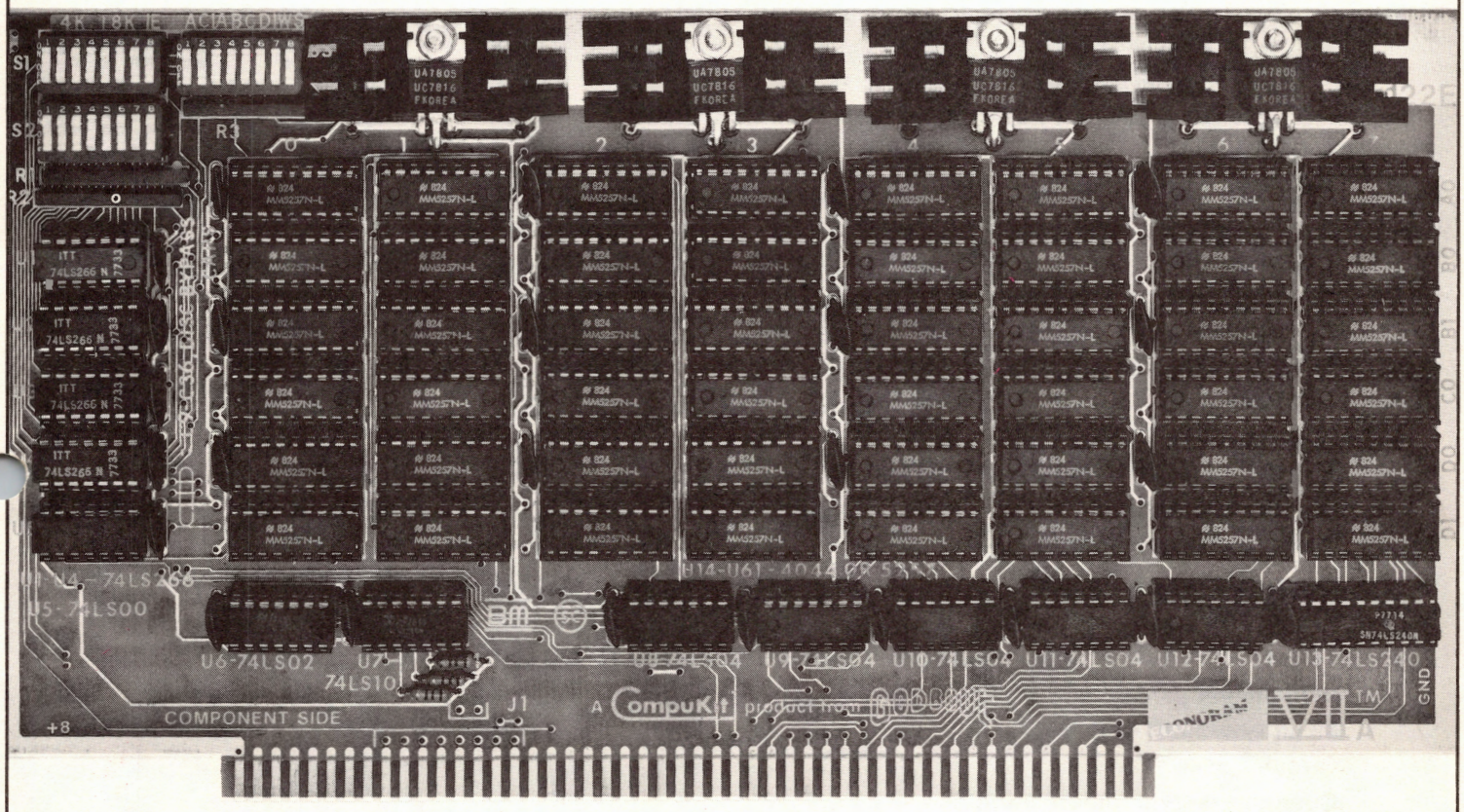
A1	8224
A2	74LS00
A3	74LS02
A4	7474
A7	8080A
A9	Data Bus Socket
A10	8212
A5	74ls04
R1 thru R13 R15 thru R17 R19 thru R21	1K ¼W
R14 R18 R25 thru R32	4.7K ¼W
R22	470 ½W
CR2	IN751
CR1	IN914
C1 C5 C7 C9 C10	33mF
C4 C6 C8 C18 C11 thru C16	.1mF
C2	39pF
C17	56pF
B2 B4 thru B7 B10	8T97
B3	74LS04
B8 B9	8216

IMS ASSOCIATES INC.
SCHEMATIC DIAGRAM
MPU-A REV. 4 2/76
©1976


B05
8080A
B224
A05
B08
B07
B09
8212
B01
B02
C01
C02
C03
J1
74LS00
74LS02
74LS04
74LS08
74LS20
BT97
BT87
7805
7812
PRESET
PRDY
XRDY
DBSTL
RESIN
RDYIN
SYNC
TANK
COM
RESET
READY
HOLD
INT
SYNC
WR
DBIN
INTE
HLDA
WAIT
SINTA
SWO
SSTACK
SHLTA
SOUT
SMI
SIOWR
SMEMR
STAT DSBL
SYNC DSBL
PHOLD
PINT
ADDR DSBL
PSYNC
PWR
PDBIN
PINTE
PHLDA
PWAIT
CCDSBL
+16 UNREG
+8 UNREG
-16 UNREG
GDM
RUN
SS
D1
IN914
HEAT SINK
LOWEST
HIGHEST
SPARES
NOTES:
1. RESISTORS ARE RATED 1/4 W, 5%
2. CAPACITORS ARE RATED
0.1 µf, 50 V, 33 µf, 20V
3. LAST REF. DES. USED
C33, R1, J1, Q4, Y1
DO NOT SCALE DRAWING

ECONORAM VII

ECONORAM VII A T.M.
USER'S MANUAL



24K x 8 static memory · S-100
using MM5257/TMS4044

122 A CompuKit T.M. product from GODBOUT

mac 8/78

Table of contents

About Econoram VII 3
Technical overview 3
Parts list 3
Assembly procedures 4
Component layout 5
Construction 6
Memory address configuration 7
Memory protect switches 7
Write strobe select switch 7
Memory testing 7
Phantom line 7
Caution 7
Schematics 8, 9, 10
Memory testing routine 11
Circuit description 11
Customer service information 12
Limited warranty information 12

ABOUT ECONORAM VII

Congratulations on your decision to purchase ECONORAM VII, a 24K x 8 memory board designed specifically for electrical and mechanical compatibility with the S-100 buss standard. The S-100 buss currently is one of the most popular in the industry and by far the most prolific; we believe this board, with the rest of the S-100 portion of the ECONORAM family, is one of the best memory boards available for that buss.

We recommend that the parts in this kit be checked against the parts list for completeness and that these instructions be read through carefully before starting. Completion of the assembly should take from one to four hours, depending on previous assembly experience, and upon completion, you will discover -- as thousands of satisfied ECONORAM owners have discovered -- the pleasure of using a fine memory board that just works, and works, and works.

As the first company to nationally offer memory kits to computer hobbyists, we again thank you for choosing ECONORAM VII . . . welcome to the club.

TECHNICAL OVERVIEW

This board incorporates proven static memory technology. There are currently two popular types of memory being used in products such as this: static and dynamic. Static memories are the overwhelming choice in applications where speed, complexity, ease of use, and reliability must all be considered. There is no refresh slowdown, the CPU is freed from the drudgery of caretaking the memory, and techniques such as direct memory access (DMA) are far more reliable and easier to implement.

The individual memory ICs used on this board are grouped together to form four larger blocks of memory, two 4K X 8 and two 8K X 8. The 4K blocks may be addressed on any 4K boundary, and the 8K blocks on any 8K boundary by setting the starting locations with the on-board dip switch (no jumpers required). Additional features include write protect switches for each of the four blocks; a write strobe selection switch which allows use of memory in systems with or without a front panel (MWRITE strobe); allowance for use with or without the PHANTOM line; thorough capacitor bypassing of supply lines to suppress transients plus on-board regulation and heat-sinking for reliably cool operation. All this and sockets for all ICs go onto a double-sided, solder-masked printed circuit board with a complete component-layout legend.

Parts List

Upon receipt of your kit, check your parts against the list below.

- ☐ (1) Econoram VIIA circuit board

INTEGRATED CIRCUITS (note: the following parts may have letter suffixes and prefixes along with the key numbers given below.)

- ☐ (48) MM5257or TMS4044 memories (U14 - U61)
- ☐ (1) 74LS00 nand gate (U5)
- ☐ (1) 74LS10 3-input nand gate (U7)
- ☐ (1) 74LS02 nor gate (U6)
- ☐ (5) 74LS04 hex inverters (U8 - U12)
- ☐ (1) 74LS240 TRI-STATE® inverters (U13)
- ☐ (4) 74LS266 ex-nor o.c. (U1 - U4)
- ☐ (4) 7805 positive 5V regulators (U62 - U65)

OTHER ELECTRONIC COMPONENTS

- ☐ (3) S.I.P. resistor packs (R1 - R3)*
- ☐ (3) 2.7K 1/4 watt resistors (red-violet-red; R4 - R6)
- ☐ (8) 39uF tantalum capacitors (C1 - C8)
- ☐ (25) ceramic disk bypass capacitors*

MECHANICAL COMPONENTS

- ☐ (61) low profile sockets*
- ☐ (3) dipswitch (S1 - S3)*
- ☐ (4) TO-220 heat sinks
- ☐ (4) 6-32 bolts
- ☐ (4) 6-32 lockwashers
- ☐ (4) 6-32 hex nuts
- ☐ (1) instruction booklet

*supplied already soldered to board.

ASSEMBLY PROCEDURES

Proper operation of your kit depends on good soldering technique, along with correct identification and handling of the various parts used during construction. Read this manual *thoroughly* before plugging in your soldering iron.

SOLDERING TECHNIQUES. The Econoram Board is *solder-masked.* In case you are not familiar with solder-masked boards, they are similar to standard PC boards, but are screened with a solder-resistant coating. This mask is screened over the entire board, except where there are solder connections to be made; since solder does not comfortably hold to or flow over the resist, the chances of getting a bridge between tight, adjacent traces are decidedly minimized.

However, soldering a solder-masked board requires a bit of care. All soldering is performed on the *solder* side of the board; we recommend keeping all component leads straight up at all times, not bent over as with some other types of boards (see figure 1). When soldering, bring the iron tip in at an angle, against the board pad and component lead; then feed in a tiny bit of solder at opposite ends of the lead (see figure 2). This makes a good joint with no excess solder. *NOTE: Use of any type of solder other than rosin core solder invalidates the warranty. Do not use any type of solder paste or corrosive flux under any conditions.*

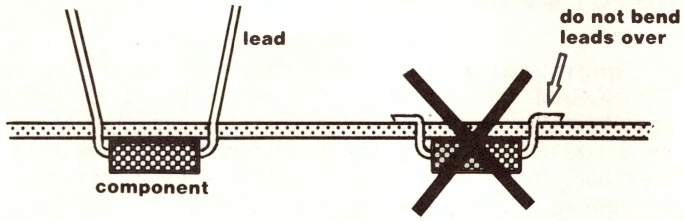


Figure 1.

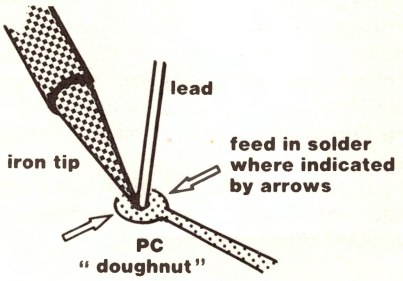


Figure 2.

IDENTIFICATION OF PARTS. There are many ICs used in this kit; each one must be oriented correctly for proper operation. Most ICs have a dot near one corner that indicates pin 1 (see figure 3); sometimes this dot appears in conjunction with a deeply cut notch or circle. Other types indicate the pin 1 end of the IC by a deep notch, or a notch within a shallow circle (see figure 4). In case of doubt, place the IC in front of you so that any identifying numbers read from left to right; pin 1 is almost always in the lower left-hand corner (figure 5).

Additionally, the 8 tantalum capacitors must be correctly identified as to the (+) and (-) ends. The (+) end is the rounded end and is also identified with a (+) mark on the body (see figure 6).

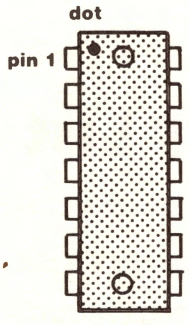


Figure 3.

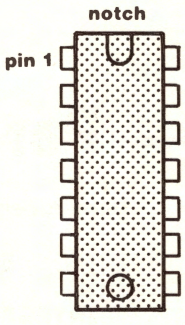


Figure 4.

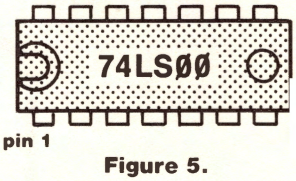


Figure 5.

HANDLING OF PARTS. All integrated circuits may be damaged by static electricity discharges; however, MOS ICs - such as the memory ICs included with your kit - are most susceptible to this problem. You can easily accumulate a static charge on your body in the thousands-of-Volts range (say, by walking across a rug on a dry day). If you then touch the pins of an IC, those thousands of Volts flow into the IC and can damage the internal structure. To prevent this from happening, leave the ICs in their protective foil until needed; then, before plugging in each IC, touch the protective foil with your finger to drain off any residual charge. Also, avoid wearing clothing that is prone to generating static electricity during construction (such as sweaters, certain types of acrylic fabrics, and so on).

Another caution concerns the mounting of resistors. These are small, somewhat fragile parts, and excessive force used while bending the leads may cause damage or crack the part. Either use a commercial lead bender (available from Godbout's and many other electronic vendors), or use needlenose pliers to grasp the lead close to the body of the resistor and then bend the lead downward (see figure 7).

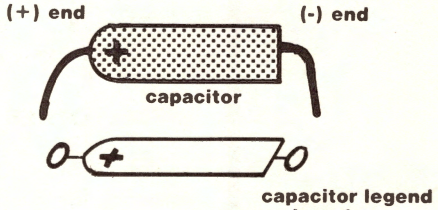


Figure 6.

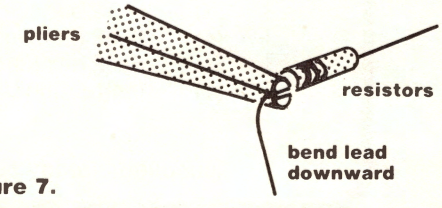
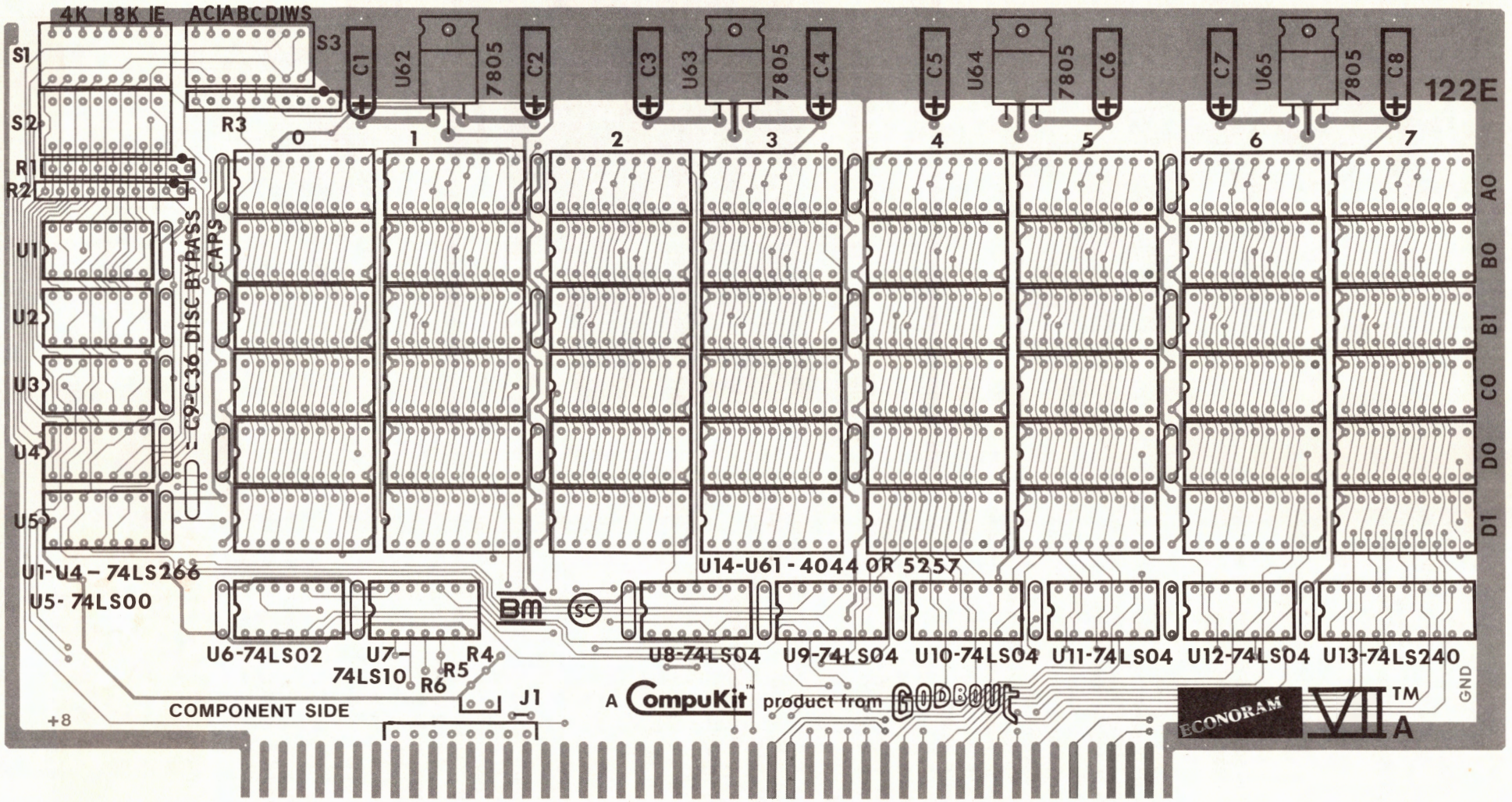


Figure 7.



Component Layout

UP

CONSTRUCTION

Keeping the preceding information in mind, it is now time to mount the various components and install the ICs into their sockets. Orient the board as shown in the component layout. Then, referring to this layout, follow the steps below in the order given.

☐ **1.** Bend leads, mount and solder the resistors in place as indicated.

☐ **2.** Bend leads, mount and solder the 8 tantalum capacitors in place as indicated, carefully observing the orientation of the (+) end.

☐ **3.** Mount the 4 regulators and the regulator heat sinks in place. Referring to Figure 8, observe that the heat sinks mount on top of the board, and then, the regulators mount on top of the heat sinks. If desired, a small dab of heat sink compound may be added between the regulator cases and heat sinks to improve thermal transfer from regulator to sink. Again referring to Figure 8, bend all regulator leads as shown, then bolt the heat sink regulator assemblies into place using the provided nut, bolt, and lockwasher hardware. After these assemblies are in place, solder the regulator leads on the solder side of the board.

☐ **4.** Observe the 8 pole dip switches. The side with the number 1-8 faces the upper edge of the board (see Figure 9). Mount the switch in place, then solder. *(Note: may be already soldered in place.)*

☐ **5.** Before proceeding, check your work so far by testing the board for any shorts between supply lines and ground. Referring to Figure 10, use a voltmeter to test for a shorted condition at the test points indicated on the regulator.

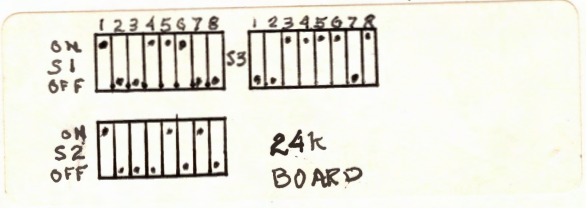
Perform this test at each regulator. If you find a short circuit at any of these points, carefully check over your board for the cause of the short. If all readings are satisfactory, proceed with construction of the board.

☐ **6.** Prepare to plug in all ICs in accordance with the component layout, Page 5. All ICs should have the pin 1 end facing towards the left side of the board. Any numbers or other markings on the IC should be right side up with the board oriented as shown in the component layout. If any of the writing is upside-down, the IC may not be correctly inserted.

A common problem with boards returned for repair is improper insertion of the IC. Sometimes a lead from the IC will bend *under* the part instead of going into the socket contacts. We recommend inserting each IC halfway, and verifying visually that all pins are indeed going into the socket. Then, push the IC in the rest of the way.

Keeping the preceding in mind, insert the support ICs first and then insert all memory ICs.

☐ **7.** Check your work over carefully for errors in construction, improperly inserted ICs, or improper polarity of the tantalum capacitors. If all is well, assembly of your board is now complete.



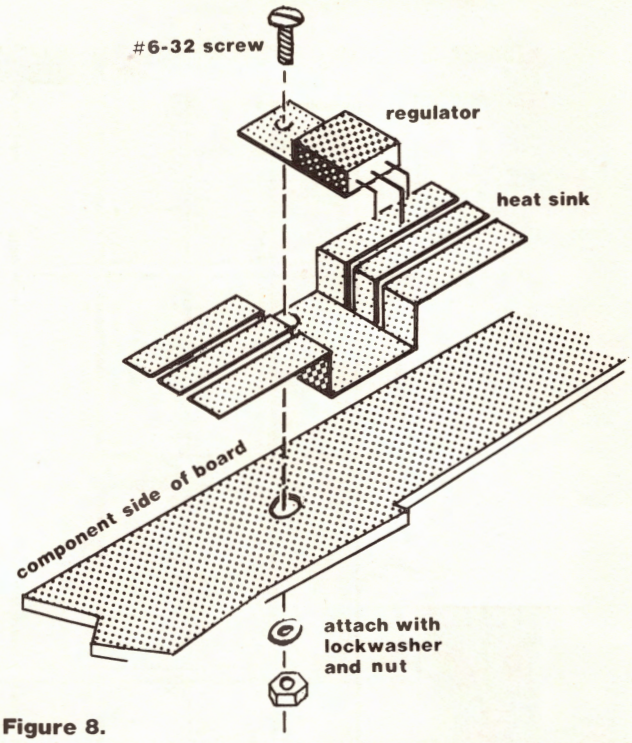


Figure 8.

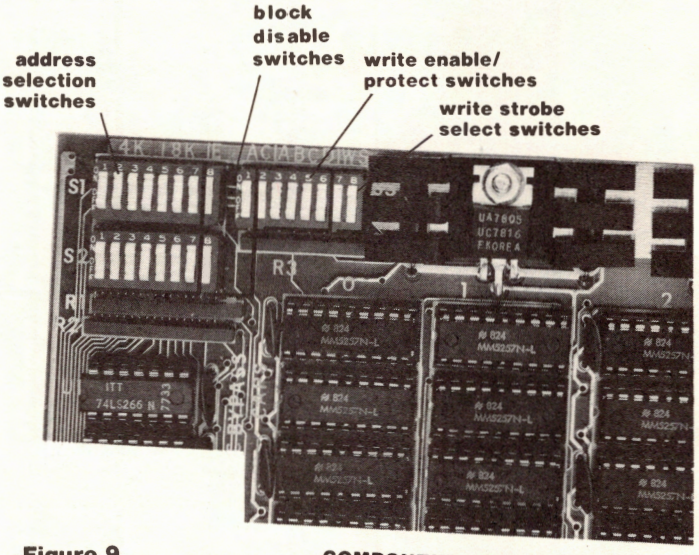


Figure 9.

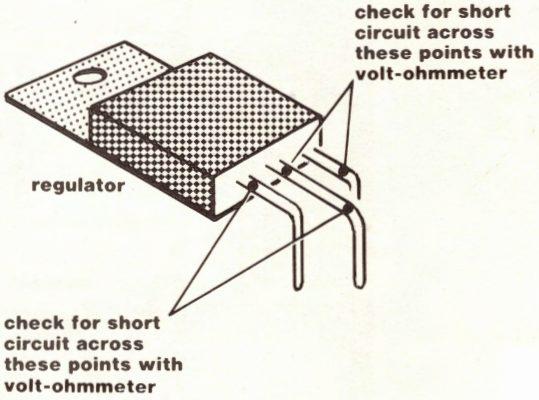


Figure 10.

MEMORY ADDRESS CONFIGURATION

This board is configured as two 4K blocks addressable on 4K boundaries and two 8K blocks independently addressable on 8K boundaries. Switches S1 and S2 each select one 4K and one 8K block. (switch S1 : block A - 4K and block B - 8K, switch S2 : block C - 4K and block D - 8K) as follows:

4K		8K	
SWITCH POSITION	STARTING ADDRESS	SWITCH POSITION	STARTING ADDRESS
1 2 3 4		5 6 7	
Ø Ø Ø Ø	ØØØØ	Ø Ø Ø	ØØØØ
Ø Ø Ø 1	1ØØØ	Ø Ø 1	2ØØØ
Ø Ø 1 Ø	2ØØØ	Ø 1 Ø	4ØØØ
Ø Ø 1 1	3ØØØ	Ø 1 1	6ØØØ
Ø 1 Ø Ø	4ØØØ	1 Ø Ø	8ØØØ
Ø 1 Ø 1	5ØØØ	1 Ø 1	AØØØ
Ø 1 1 Ø	6ØØØ	1 1 Ø	CØØØ
Ø 1 1 1	7ØØØ	1 1 1	EØØØ
1 Ø Ø Ø	8ØØØ		
1 Ø Ø 1	9ØØØ		
1 Ø 1 Ø	AØØØ		
1 Ø 1 1	BØØØ		
1 1 Ø Ø	CØØØ		
1 1 Ø 1	DØØØ		
1 1 1 Ø	EØØØ		
1 1 1 1	FØØØ		

Ø = OFF

1 = ON

In addition, position 8 on each switch enables (OFF) or disables (ON) the associated 8K block. Positions 1 and 2 of switch S3 disable/enable the 4K blocks A and C respectively.

MEMORY PROTECT SWITCHES

Switch S3 positions 3-6 are write enable switches. Conversely they may be used for manual write protection of the memory. Each position 3, 4, 5 and 6 is associated wih one memory block A, B, C and D respectively. Any combination of these four swtiches may be ON write enabling or OFF write protecting the particular block.

WRITE STROBE SELECT SWITCHES

Switch S3 positions 7 and 8 select write stobe. S3-7 ON causes $\overline{PRW}$ to quality the memory for write commands. S3-8 ON causes MWRITE to qualify the memory for write commands. In normal operations with systems which have front panels ie., Altair, Imsai etc., or others which generate $\overline{MWRITE}$ with or without front panels use MWRITE. All others use $\overline{PWR}$. Positions 7 and 8 both ON at the same time will ground MWRITE on the buss. This condition must be avoided if MWRITE is present.

MEMORY TESTING

If the memory board seems to be working properly, the Memory Testing Routine (page 11) can be used to give the board a more thorough workout. It is rather slow; but will do the job well. It can be entered via editor/assembler or front panel switches.

The routine is set up to test 24K from 4000 hex up to AØØØ hex. This may be changed by entering a different starting address at "STRT" (3001 - 3002) and/or a different end address at "END" (3004 - high order byte only).

If the memory passes the test it starts over again. You may on the other hand, insert a jump instruction at "MARK" to some user routine or, if desired the user may enter an output instruction or, can do a notification routine at "MARK" to show successful completion and restart.

If the memory fails the test, critical information is stored and the routine enters a software "HALT", that is a "jump to here' at "SHLT". Front panel lights, if any, will show this state. The user may then use the front panel or dump routines to display the following stored failure info:

3069*	"FDE"	= D, E pair . . . D is the fill character and E is the test character
306B*	"FHL"	= H, L pair . . . the failure address
306D*	"FOUT"	= the data expected at this address
306E*	"FIN"	= the data read from, the address

* address from *Memory Testing Routine Listing.*

The user may replace the "jump" at "SHLT" with a jump to a display or notification routine.

The difference between "FOUT" and "FIN" should indicate which bit is failing, indicating which chip or area is causing the problem.

This test will find most of the harder to distinguish errors.

PHANTOM LINE

In response to increasing numbers of users who have requested inclusion of "PHANTOM LINE", buss pin 67 which is often used for implementing power on jump features. This board is designed for use with active or inactive PHANTOM lines.

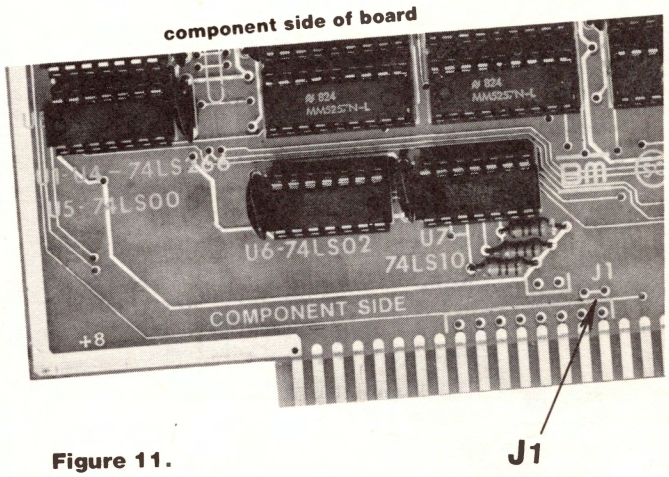


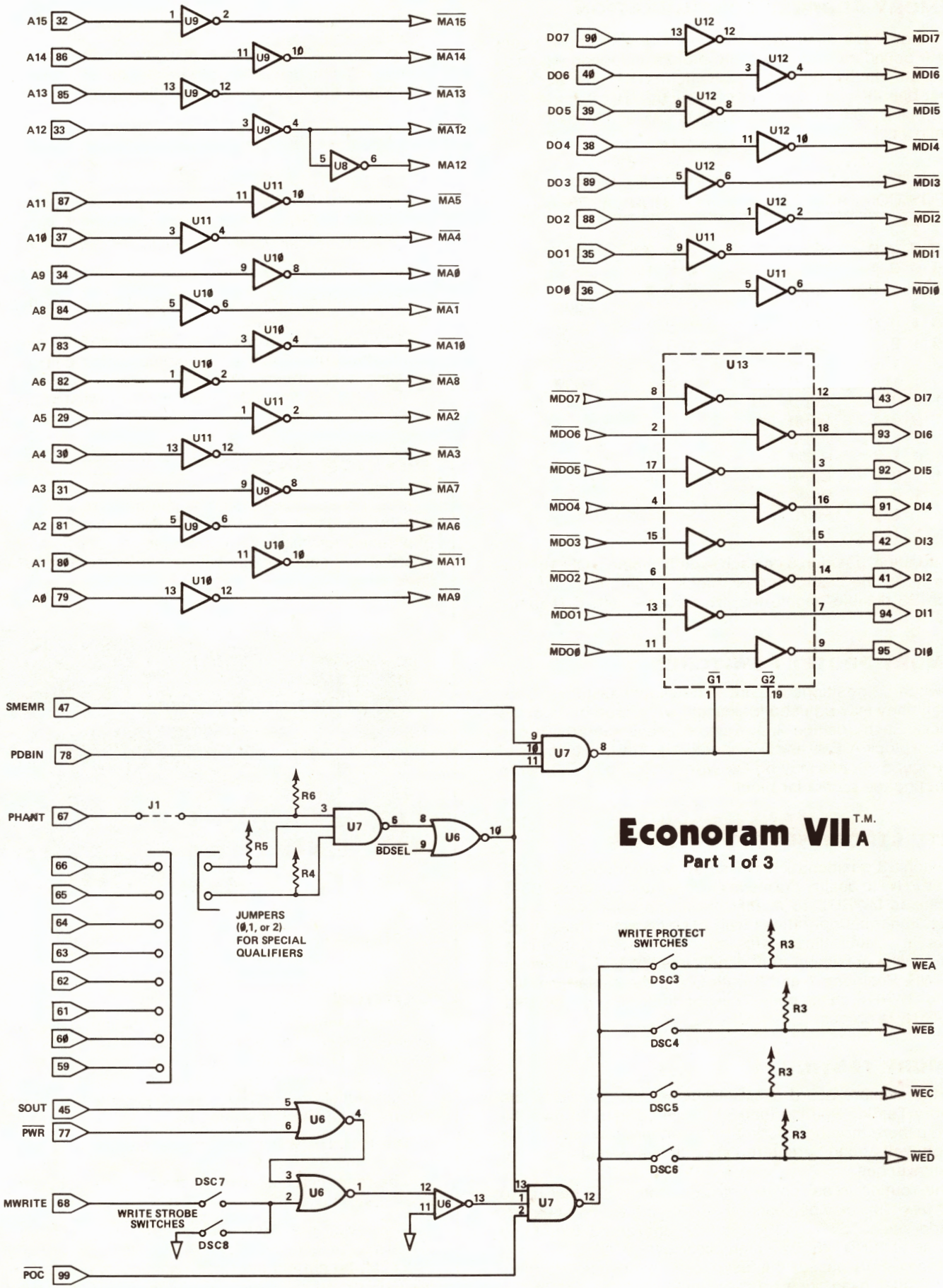
Figure 11.

CAUTION

Some manufacturers use PHANTOM line (buss pin 67) for a refresh signal. This will conflict with the PHANTOM feature, and cause boards with PHANTOM to fail. If your system has this conflict it must be resolved by either eliminating the refresh signal on the CPU board or disabling the PHANTOM feature on this and other boards.

The PHANTOM feature may be disabled by cutting the trace on the back of the PC board between pads for J1. It may be reactivated at any time simply by installing a jumper at J1 (see Figure 11).

If you want the PHANTOM feature, the conficting refresh signal may be eliminated (IF NOT USED ELSEWHERE IN THE SYSTEM) by cutting the trace connected to buss pin 67 on the CPU board. *BE SURE OF YOUR SYSTEM CONFIGURATION BEFORE CUTTING ANY TRACES.*



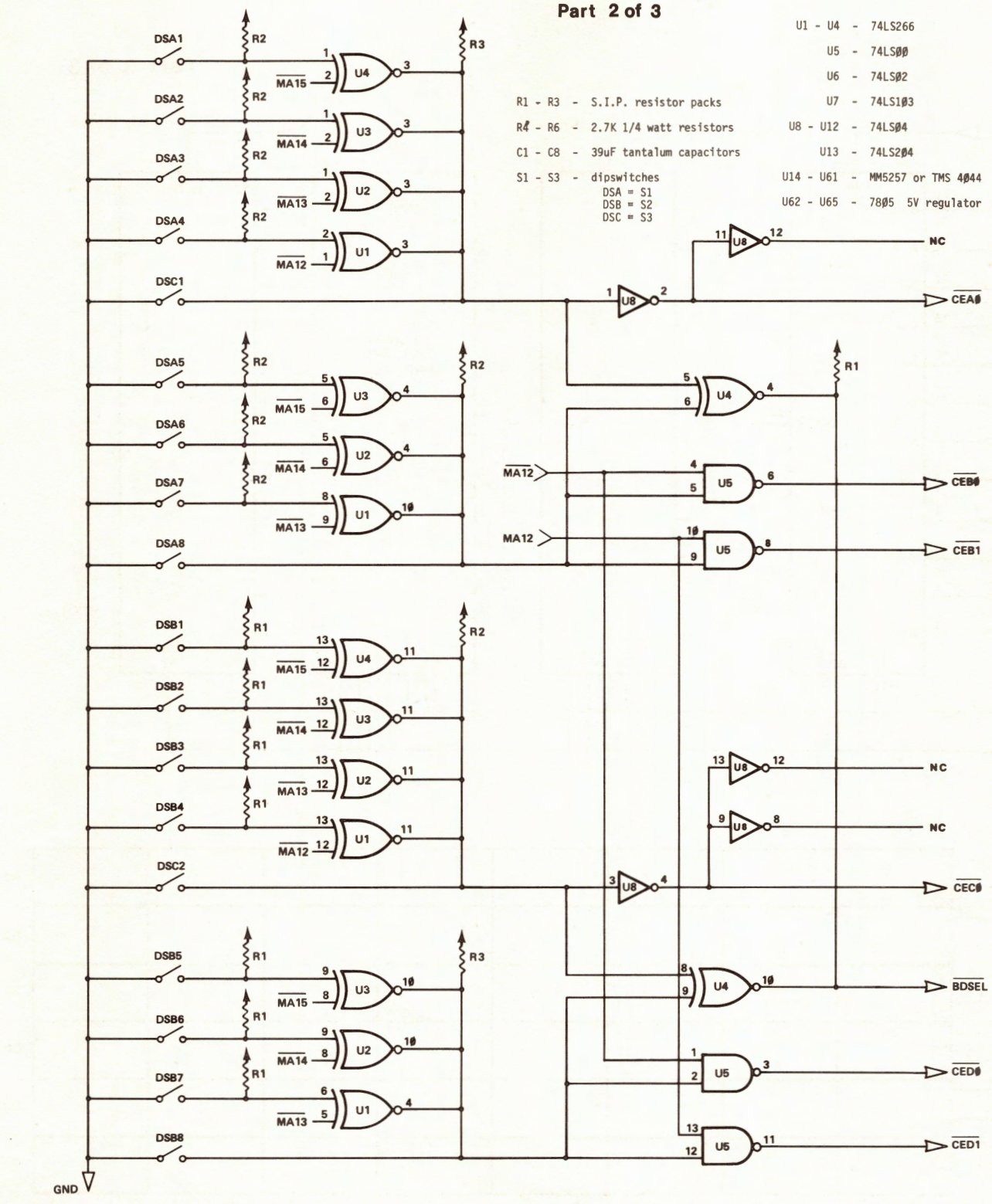
Econoram VII^{A} T.M.

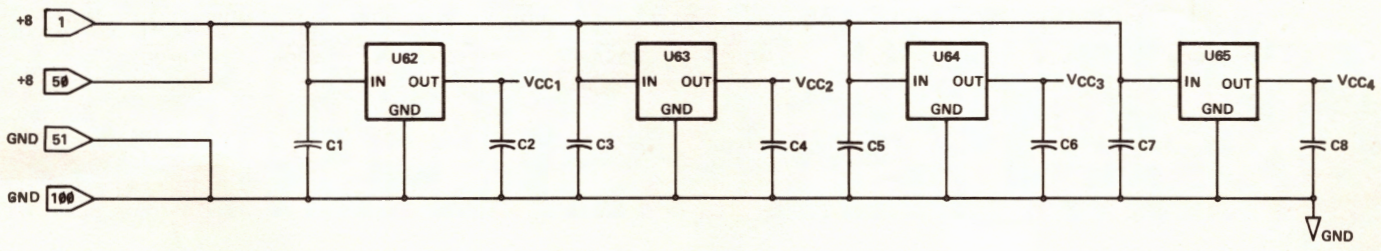
Part 1 of 3

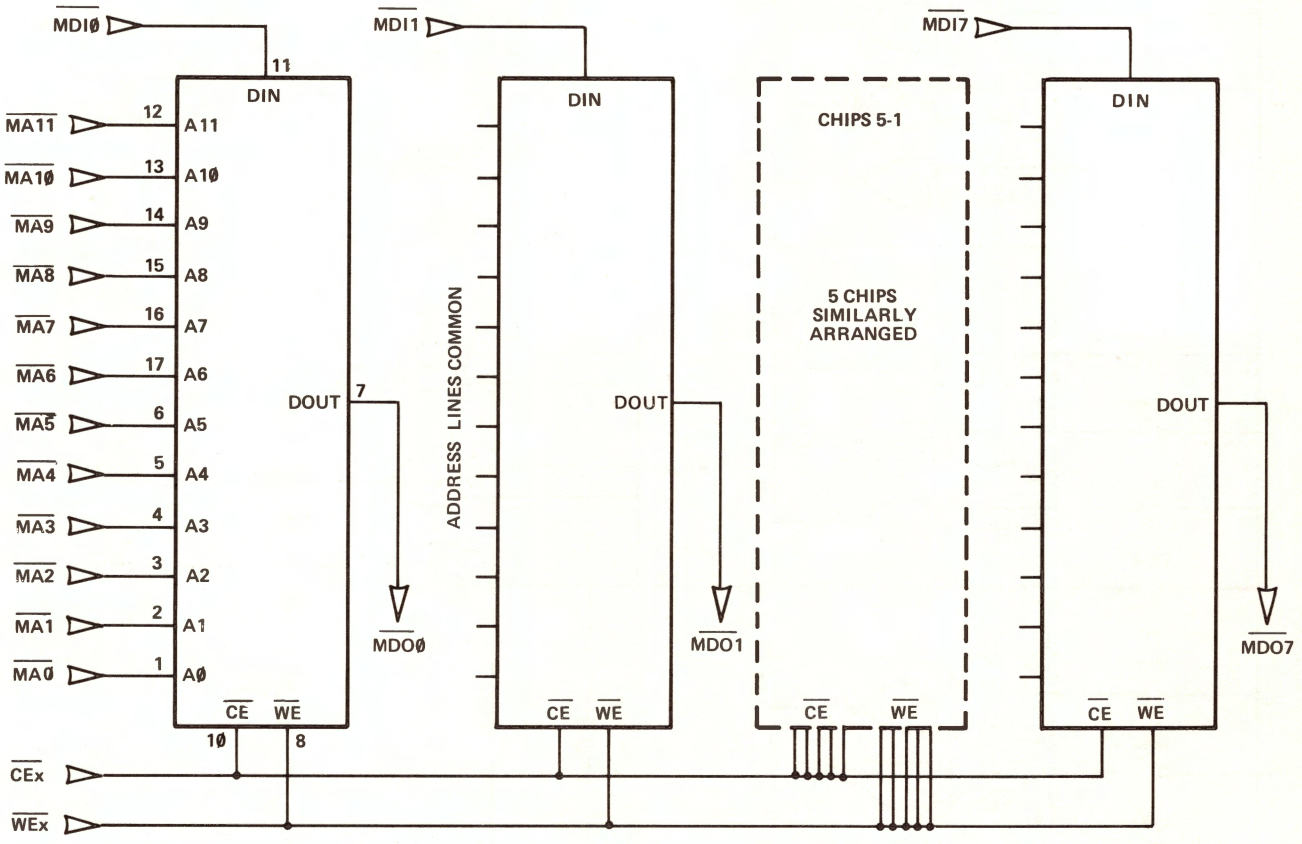
Part 2 of 3

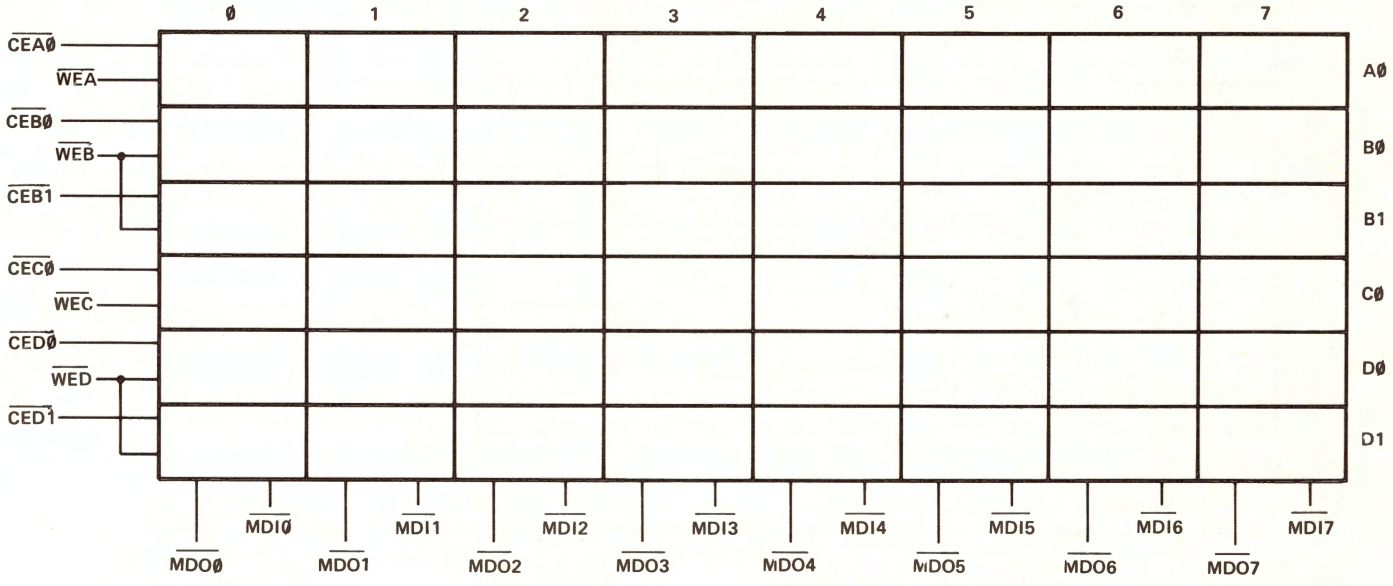
R1 - R3 - S.I.P. resistor packs
R4 - R6 - 2.7K 1/4 watt resistors
C1 - C8 - 39uF tantalum capacitors
S1 - S3 - dipswitches
DSA = S1
DSB = S2
DSC = S3

U1 - U4 - 74LS266
U5 - 74LSØØ
U6 - 74LSØ2
U7 - 74LS1Ø3
U8 - U12 - 74LSØ4
U13 - 74LS2Ø4
U14 - U61 - MM5257 or TMS 4Ø44
U62 - U65 - 78Ø5 5V regulator






MDI0
MDI1
MDI7
11
DIN
DIN
DIN
CHIPS 5-1
5 CHIPS
SIMILARLY
ARRANGED
ADDRESS LINES COMMON
MA11
MA10
MA9
MA8
MA7
MA6
MA5
MA4
MA3
MA2
MA1
MA0
12
13
14
15
16
17
6
5
4
3
2
1
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
DOUT
7
MDO0
MDO1
MDO7
CE
WE
10
8
CEx
WEx


0
1
2
3
4
5
6
7
CEA0
WEA
CEB0
WEB
CEB1
CEC0
WEC
CED0
WED
CED1
A0
B0
B1
C0
D0
D1
MDI0
MDI1
MDI2
MDI3
MDI4
MDI5
MDI6
MDI7
MDO0
MDO1
MDO2
MDO3
MDO4
MDO5
MDO6
MDO7

MEMORY TESTING ROUTINE

Figure 13.

```
 3000 21 00 40          0010 STRT LXI  H,4000H
 3003 3E A0             0020 END  MVI  A,0A0H
3005 32 6E 30           0030      STA  FIN
3008 3E 10              0040      MVI  A,10H
300A 84                 0050      ADD  H
300B 4F                 0060      MOV  C,A
300C 16 00              0070      MVI  D,0
300E 1E FF              0080      MVI  E,0FFH
3010 22 65 30           0090 DONE SHLD STAD
3013 AF                 0100      XRA  A
3014 47                 0110      MOV  B,A
3015 7B                 0120 SCND MOV  A,E
3016 5A                 0130      MOV  E,D
3017 57                 0140      MOV  D,A
3018 79                 0150      MOV  A,C
3019 2A 65 30           0160      LHLD STAD
301C 72                 0170 FILL MOV  M,D
301D 23                 0180      INX  H
301E BC                 0190      CMP  H
301F C2 1C 30           0200      JNZ  FILL
3022 2A 65 30           0210      LHLD STAD
3025 73                 0220 NEXT MOV  M,E
3026 7B                 0230      MOV  A,E
3027 BE                 0240      CMP  M
3028 C2 6F 30           0250      JNZ  FAIL
302B 79                 0260      MOV  A,C
302C 23                 0270      INX  H
302D 94                 0280      SUB  H
302E C2 4D 30           0290      JNZ  NDON
3031 B8                 0300      CMP  B
3032 44                 0310      MOV  B,H
3033 CA 15 30           0320      JZ   SCND
3036 3A 66 30           0325      LDA  STAD+1
  39 00                 0330 MARK NOP
  3A 00                 0331      NOP
  3B 00                 0332      NOP
303C 3A 6E 30           0340      LDA  FIN
303F B9                 0350      CMP  C
3040 CA 00 30           0360      JZ   STRT
3043 79                 0370      MOV  A,C
3044 67                 0380      MOV  H,A
3045 2E 00              0390      MVI  L,0
3047 C6 10              0400      ADI  10H
3049 4F                 0410      MOV  C,A
304A C3 10 30           0420      JMP   DONE
304D 22 67 30           0430 NDON SHLD NXAD
3050 7A                 0440 LOPB MOV  A,D
3051 BE                 0450 LOPA CMP  M
3052 C2 6F 30           0460      JNZ  FAIL
3055 2C                 0470      INR  L
3056 C2 51 30           0480      JNZ  LOPA
3059 79                 0490      MOV  A,C
305A 24                 0500      INR  H
305B BC                 0510      CMP  H
305C C2 50 30           0520      JNZ  LOPB
305F 2A 67 30           0530      LHLD NXAD
3062 C3 25 30           0540      JMP  NEXT
3065                    0550 STAD DS   2
3067                    0560 NXAD DS   2
3069                    0570 FDE  DS   2
306B                    0580 FHL  DS   2
306D                    0590 FOUT DS   1
306E                    0600 FIN  DS   1
306F 22 6B 30           0610 FAIL SHLD FHL
3072 32 6D 30           0620      STA  FOUT
3075 7E                 0630      MOV  A,M
3076 32 6E 30           0640      STA  FIN
 179 EB                 0650      XCHG
 17A 22 69 30           0660      SHLD FDE
 17D C3 7D 30           0670 SHLT JMP SHLT
3080                    0680 *
```

CIRCUIT DESCRIPTION

The heart of Econoram VII is the MM5257/TMS4044 static memory IC (RAM), which can store 4096 single bits of information (thus, each is a "4K x 1" memory IC). Unlike standard RAMs, those included with your kit are specifically designated by the manufacturer as low power, high speed parts.

These ICs are arranged in rows that are 8 ICs wide. This way, each row can store 4K x 8 bits of information. Paralleling 6 of these rows together produces a total memory storage of 24K x 8 bits. (Note that the bit number corresponding to a given column of ICs is indicated along the top edge of the memory array).

Now that we have this storage, there are still other aspects we must consider. First, we need to address a specific location in memory; and, we need to be able to *write* data into the memory, or *read* data from the memory.

The schematics on pages 8 and 9 show the address circuitry along with the other Econoram VII circuitry. Each memory IC requires 12 address bits (A0-A11) to access any one of the 4096 bits available in the IC. These address bits are generated by the CPU and are buffered by a number of inverters. After buffering, a particular address is presented to all IC address selection pins. However, we additionally need to select which particular row of ICs is to react to the given address. This requires 4 more address bits (A12-A15), which are decoded and used to enable the desired row of ICs (note row markings along the right hand side of the memory array).

When data is to be written into memory, it first passes through 8 inverting buffers before being put on the data pins of the RAMs (buffering prevents loading of the data buss). Data to be read on to the data buss from memory passes through 8 TRI-STATE® inverting buss drivers; when data is not being read on to the buss, the outputs of these inverters are in a high-impedance or "disconnected" state.

An unfortunate fact of life is that logic ICs generate switching transients that travel along the power supply lines. If these transients work their way into the logic circuitry, problems can appear. To prevent such occurances, bypass capacitors are tied across the power lines at regular intervals in the memory array and at every support IC.

This board is guaranteed to operate at 2 MHZ over the full temperature range (0 - 70 C ambient) and to draw less than 2500 mA (2.5 amps). However, we have tested many of them successfully at 4 MHz with a Z-80 though no instruction fetches were involved in the test. Also, our typical measured currents were less than 2100 mA at cold start-up, rapidly decreasing to around 1600-1900 mA, depending on the surrounding temperature. We have heard similar reports from the people already using these boards.

It is interesting to note that static RAM technology has progressed to the point that this high-performance static RAM board is comparable in cost and power consumption to dynamic memory boards.

THANK YOU

This board is the result of much time, work and experience on the part of a number of people. In addition to thanking you for choosing this board, we'd like to thank Spencer Cottam, Bruce Mycroft, Mac Mullen, Pam Martin and many others for their help with this project.

We strive for a board that doesn't just work the first time, but continues to give reliable operation for a long time. If we can be of any help to you in applying this board, or if you have any questions, please let us know. As always, we solicit your comments, letters and new product suggestions.

HAPPY COMPUTING!

CUSTOMER SERVICE INFORMATION

Our paramount concern is that you be satisfied with any Godbout CompuKit product. If this product fails to operate properly, it may be returned to us, see warranty information below.

If you have any questions about assembly, performance, specifications or need further information feel free to write us at:

P.O. Box 2355, Oakland Airport, CA 94614.

When writing, please be as specific as possible concerning the nature of your query. We maintain a 24 hour a day phone, for taking orders, (415) 562-0636. If you have problems or questions which cannot be handled by mail, this number can be used to connect you with our technical people ONLY during normal business hours (10am-5pm Pacific Time). Unfortunately, we cannot return calls, or accept collect calls.

LIMITED WARRANTY INFORMATION

Godbout Electronics will repair or replace, at our option, any parts found to be defective in either materials or workmanship for a period of 1 year from date of invoice. Defective parts *must* be returned for replacement.

If a defective part or design error causes a Godbout Electronics product to operate improperly during the 1 year warranty period, we will service it free (original owner only) if delivered and shipped at owner's expense to Godbout Electronics. If improper operation is due to an error or errors on the part of the purchaser, there may be a repair charge. Purchaser will be notified if this charge exceeds $10.00.

We are not responsible for damage caused by use of solder intended for purposes other than electronic equipment construciton, failure to follow printed instructions, misuse or abuse, unauthorized modifications, use of our products in applications other than those intended by Godbout Electronics, theft, fire, or accidents.

Return to purchaser of a fully functioning unit meeting all advertised specifications in effect as of date of purchase is considered to be complete fulfillment of all warranty obligations assumed by Godbout Electronics. This warranty covers only products marketed by Godbout Electronics and does not cover other equipment used in conjunction with said products. We are not responsible for incidental or consequential damages.

Prices and specifications are subject to change without notice, owing to the volatile nature and pricing structure of the electronics industry.

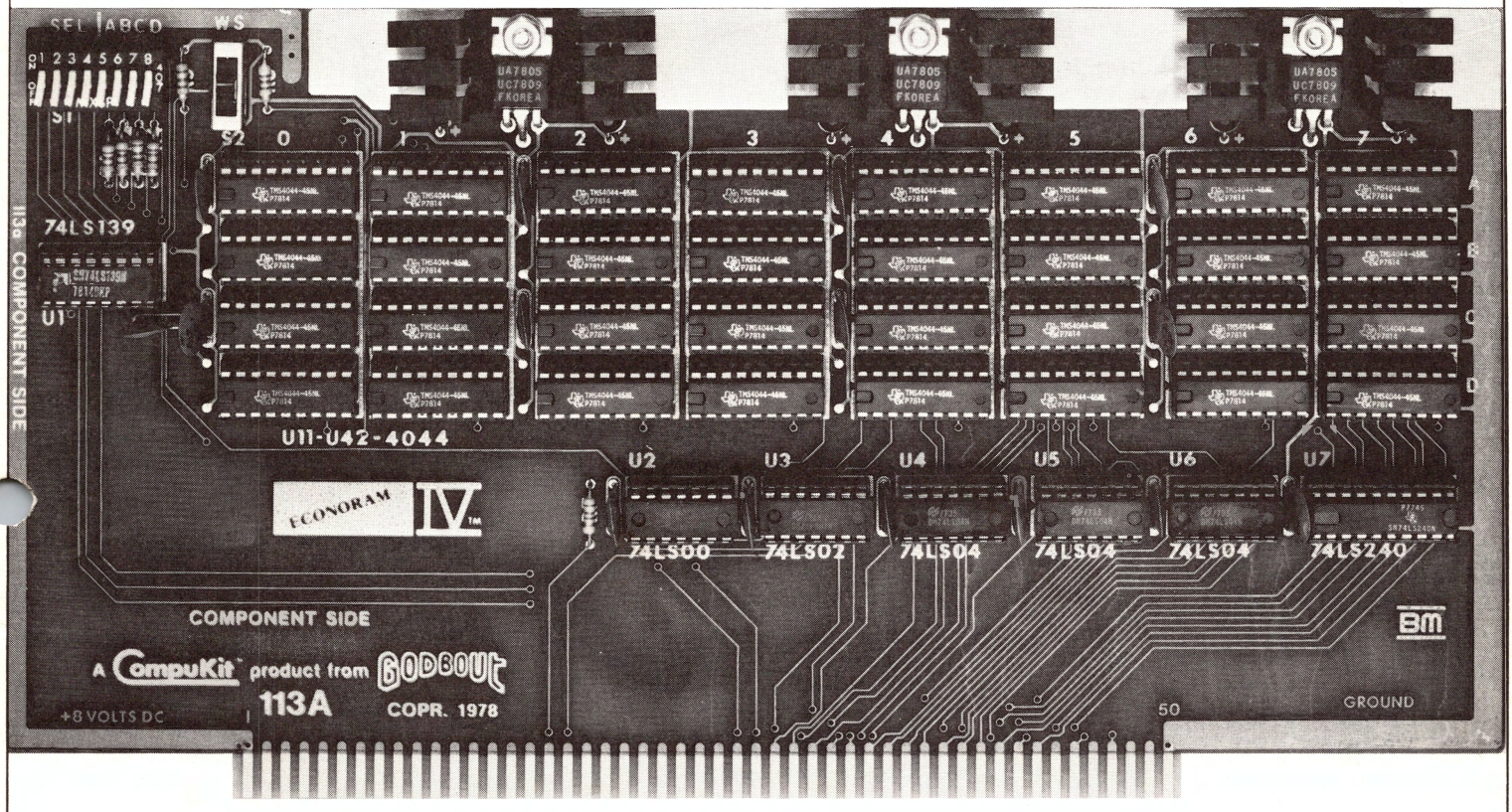
"Econoram" and "Econoram VII" are trademarks of W.J. Godbout.

TRI-STATE® is a trademark of National Semiconductor Corp.

Contents of this booklet ©1978 by Godbout Electronics. All rights reserved. We encourage quotation for the purposes of product review if source is credited.

ECONORAM IV

ECONORAM IV T.M.
USER'S MANUAL



16K x 8 static memory • S-100
using TMS40L44/MM5257 • 4MHz

113 mac

A CompuKit T.M. product from GODBOUT

Rev 3/79

Table of contents

About Econoram IV. 3
Technical overview 3
Parts list 3
Assembly procedures 4
Component layout 5
Construction 6
Memory address assignment/disable 7
Memory protect switches. 7
Write strobe select switch. 7
Phantom line. 7
Caution 7
Schematics 8, 9
Memory testing 10
Memory testing routine. 10
Circuit description 11
Customer service information. 12
Limited warranty information. 12

ABOUT ECONORAM IV

Congratulations on your decision to purchase ECONORAM IV, a 16K x 8 memory board designed specifically for electrical and mechanical compatibility with the S-100 buss standard. The S-100 buss currently is one of the most popular in the industry and by far the most prolific; we believe this board, with the rest of the S-100 portion of the ECONORAM family, is one of the best memory boards available for that buss.

We recommend that the parts in this kit be checked against the parts list for completeness and that these instructions be read through carefully before starting. Completion of the assembly should take from one to four hours, depending on previous assembly experience, and upon completion, you will discover -- as thousands of satisfied ECONORAM owners have discovered -- the pleasure of using a fine memory board that just works, and works, and works.

As the first company to nationally offer memory kits to computer hobbyists, we again thank you for choosing ECONORAM IV . . . welcome to the club.

TECHNICAL OVERVIEW

This board incorporates proven static memory technology. There are currently two popular types of memory being used in products such as this: static and dynamic. Static memories are the overwhelming choice in applications where speed, complexity, ease of use, and reliability must all be considered . . . there is no refresh slowdown, the CPU is freed from the drudgery of caretaking the memory, and techniques such as direct memory access (DMA) are far more reliable and easier to implement.

The individual memory ICs used on this board are grouped together to form a single 16K x 8 block of memory, addressable on any 16K boundary using the on-board dip switch (no jumpers required). Additional features include write protect switches for each 4K sector of the 16K block; a write strobe selection switch which allows use of memory in systems with or without a front panel (MWRITE strobe); allowance for use with or without the PHANTOM line; thorough capacitor bypassing of supply lines to suppress transients; plus on-board regulation and heat-sinking for reliably cool operation. All this and sockets for all ICs go onto a double-sided, solder-masked printed circuit board with a complete component-layout legend.

Parts List

Upon receipt of your kit, check your parts against the list below.

- ☐ (1) Econoram IV circuit board.

INTEGRATED CIRCUITS (note: the following parts may have letter suffixes and prefixes along with the key numbers given below.)

- ☐ (32) TMS4044 or MM5257N-3L memories (U11 - U42)
- ☐ (1) 74LS00 nand gate (U2)
- ☐ (1) 74LS02 nor gate (U3)
- ☐ (3) 74LS04 hex inverters (U4 - U6)
- ☐ (1) 74LS240 TRI-STATE® inverters (U7)
- ☐ (1) 74LS139 dual 1 of 4 decoder (U1)
- ☐ (3) 7805 positive 5V regulators (U8 - U10)

OTHER ELECTRONIC COMPONENTS

- ☐ (7) 2.7K ohm resistors (red-violet-red; R1 - R7)
- ☐ (6) 39uF tantalum capacitors (C1 - C6)
- ☐ (14) ceramic disk bypass capacitors*

MECHANICAL COMPONENTS

- ☐ (39) low profile sockets*
- ☐ (1) slide switch (S2)
- ☐ (1) 8 pole dip switch (S1)
- ☐ (3) heat sinks for regulator ICs
- ☐ (3) 6-32 bolts
- ☐ (3) 6-32 lockwashers
- ☐ (3) 6-32 nuts
- ☐ (1) instruction booklet

*supplied already soldered to board.

ASSEMBLY PROCEDURES

Proper operation of your kit depends on good soldering technique, along with correct identification and handling of the various parts used during construction. Read this manual *thoroughly* before plugging in your soldering iron.

SOLDERING TECHNIQUES. The Econoram Board is *solder-masked.* In case you are not familiar with solder-masked boards, they are similar to standard PC boards, but are screened with a solder-resistant coating. This mask is screened over the entire board, except where there are solder connections to be made; since solder does not comfortably hold to or flow over the resist, the chances of getting a bridge between tight, adjacent traces are decidedly minimized.

However, soldering a solder-masked board requires a bit of care. All soldering is performed on the *solder* side of the board; we recommend keeping all component leads straight up at all times, not bent over as with some other types of boards (see figure 1). When soldering, bring the iron tip in at an angle, against the board pad and component lead; then feed in a tiny bit of solder at opposite ends of the lead (see figure 2). This makes a good joint with no excess solder. *NOTE: Use of any type of solder other than rosin core solder invalidates the warranty. Do not use any type of solder paste or corrosive flux under any conditions.*

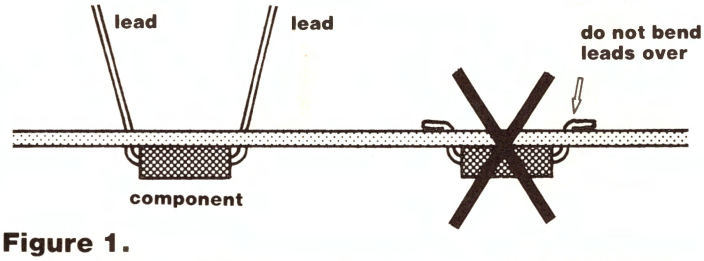


Figure 1.

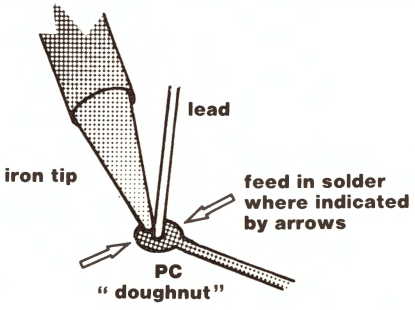


Figure 2.

IDENTIFICATION OF PARTS. There are many ICs used in this kit; each one must be oriented correctly for proper operation. Most ICs have a dot near one corner that indicates pin 1 (see figure 3); sometimes this dot appears in conjunction with a deeply cut notch or circle. Other types indicate the pin 1 end of the IC by a deep notch, or a notch within a shallow circle (see figure 4). In case of doubt, place the IC in front of you so that any identifying numbers read from left to right; pin 1 is almost always in the lower left-hand corner (figure 5).

Additionally, the 6 tantalum capacitors must be correctly identified as to the (+) and (-) ends. The (+) end is the rounded end and is also identified with a (+) mark on the body (see figure 6).

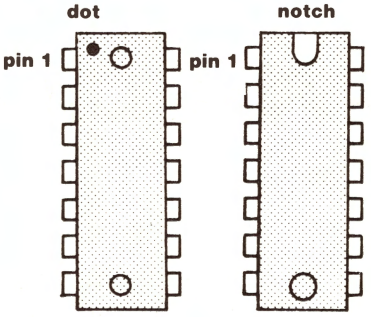


Figure 3.

Figure 4.

Figure 5.

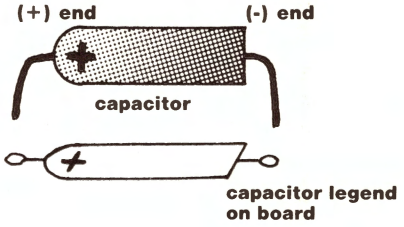


Figure 6.

HANDLING OF PARTS. All integrated circuits may be damaged by static electricity discharges; however, MOS ICs - such as the memory ICs included with your kit - are most susceptible to this problem. You can easily accumulate a static charge on your body in the thousands-of-Volts range (say, by walking across a rug on a dry day). If you then touch the pins of an IC, those thousands of Volts flow into the IC and can damage the internal structure. To prevent this from happening, leave the ICs in their protective foil until needed; then, before plugging in each IC, touch the protective foil with your finger to drain off any residual charge. Also, avoid wearing clothing that is prone to generating static electricity during construction (such as sweaters, certain types of acrylic fabrics, and so on).

Another caution concerns the mounting of resistors. These are small, somewhat fragile parts, and excessive force used while bending the leads may cause damage or crack the part. Either use a commercial lead bender (available from Godbout's and many other electronic vendors), or use needlenose pliers to grasp the lead close to the body of the resistor and then bend the lead downward (see figure 7).

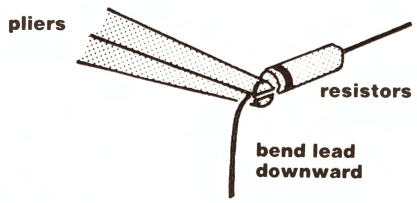
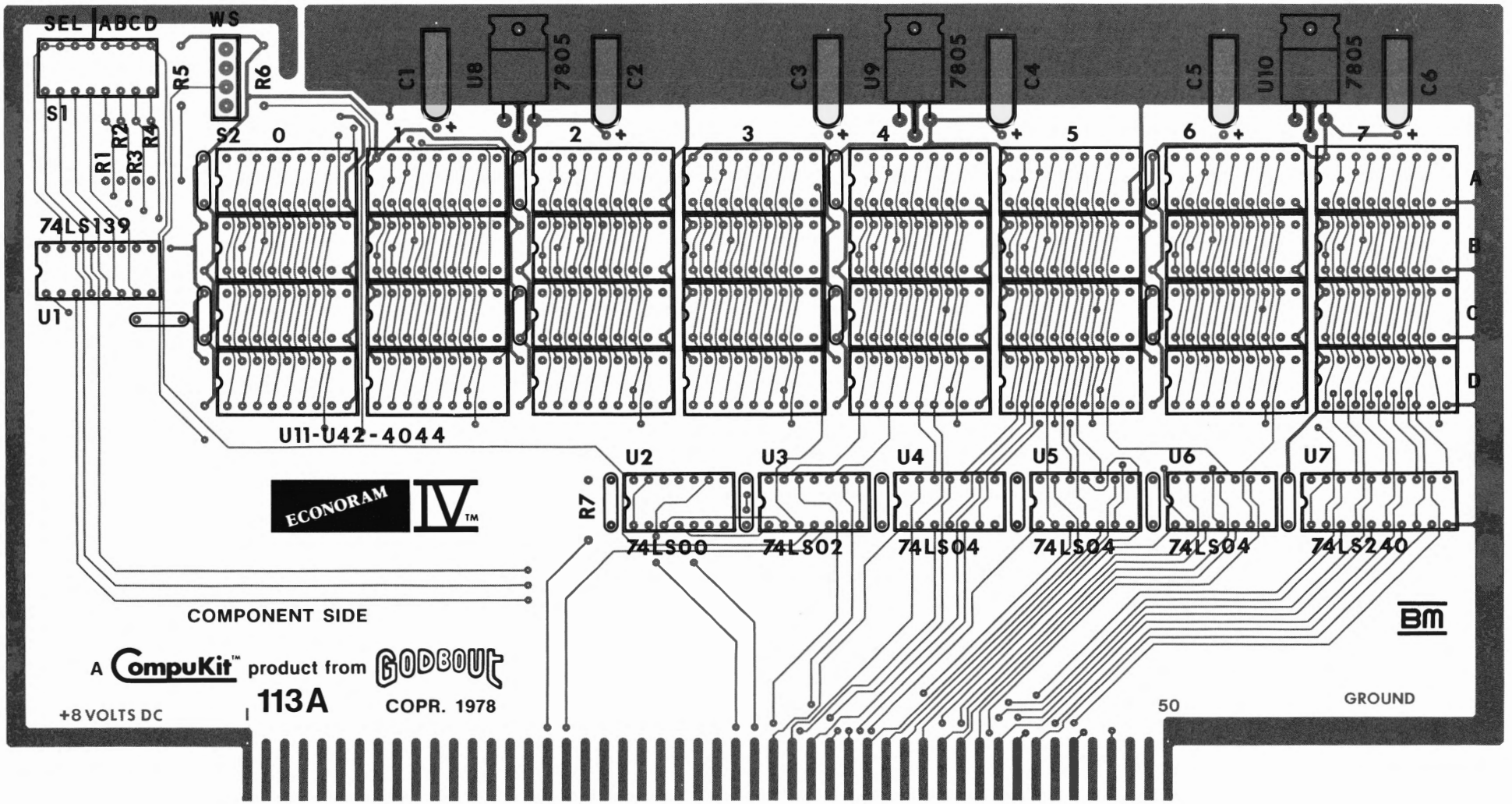


Figure 7.



Component Layout

CONSTRUCTION

Keeping the preceding information in mind, it is now time to mount the various components and install the ICs into their sockets. Orient the board as shown in the component layout; then referring to this layout, follow the steps below in the order given.

☐ **1.** Bend leads, mount, and solder the resistors in place as indicated.

☐ **2.** Bend leads, mount and solder the 6 tantalum capacitors in place as indicated, carefully observing the orientation of the (+) end.

☐ **3.** Mount the slide switch in position as shown in figure 8 (note: orientation is not important).

☐ **4.** Mount the three regulators and the regulator heat sinks in place. Referring to figure 9, observe that the heat sinks mount on top of the board, and then, the regulators mount on top of the heat sinks. If desired, a small dab of heat sink compound may be added between the regulator cases and heat sinks to improve thermal transfer from regulator to sink. Again referring to figure 9, bend all regulator leads as shown, then bolt the heat sink regulator assembles into place using the provided nut, bolt, and lockwasher hardware. After these assemblies are in place, solder the regulator leads on the non-component side of the board.

☐ **5.** Observe the 8 pole dip switch. The side with the number 1-8 faces the upper edge of the board (see figure 8). Mount the switch in place, and then solder. *(Note: May be already soldered in place.)*

☐ **6.** Before proceeding, check your work so far by testing the board for any shorts between supply lines and ground. Referring to figure 10, use a Voltmeter to test for a shorted condition at the test points indicated on the regulator.

Perform this test at each regulator. If you find a short circuit at any of these points, carefully check over your board for the cause of the short. If all readings are satisfactory, proceed with construction of the board.

☐ **7.** Prepare to plug in all ICs in accordance with the component layout, page 5. All ICs should have the pin 1 end facing towards the *left* side of the board; any numbers or other markings on the IC should be right side up with the board oriented as shown in the component layout. If any of the writing is upside down, the IC may not be correctly inserted.

A common problem with boards returned for repair is improper insertion of the IC; sometimes a lead from the IC will bend *under* the part instead of going into the socket contacts. We recommend inserting each IC halfway, and verifying visually that all pins are indeed going into the socket. Then, push the IC in the rest of the way.

Keeping the preceding in mind, insert the support ICs first, and then insert all memory IC.

☐ **8.** Check your work over carefully for errors in construction, improperly inserted ICs, or improper polarity of the tantalum capacitors. If all is well, assembly of your board is now complete.

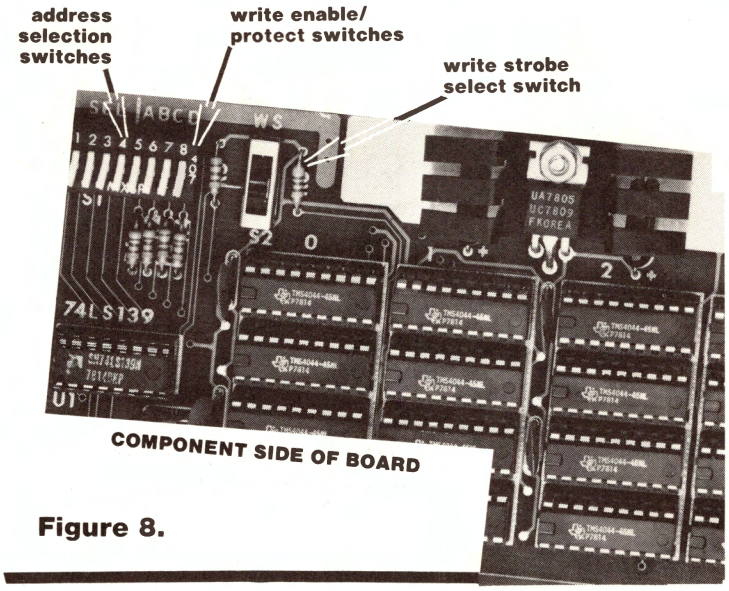


Figure 8.

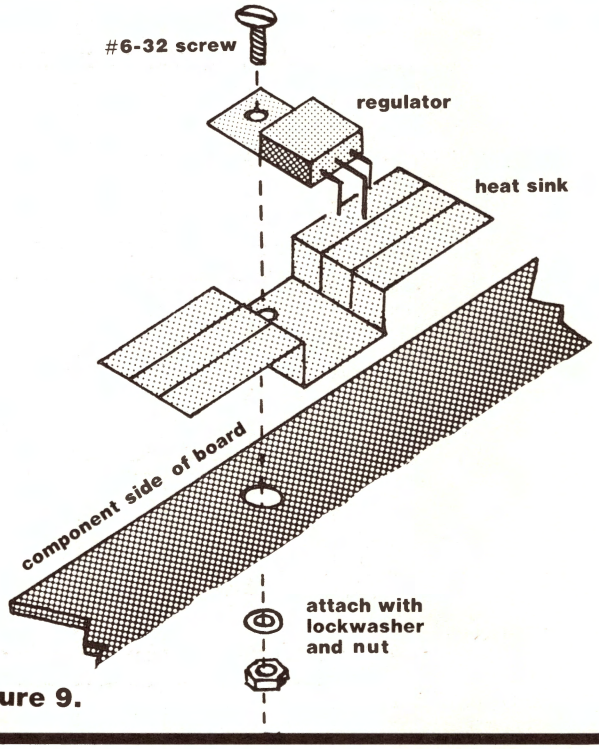


Figure 9.

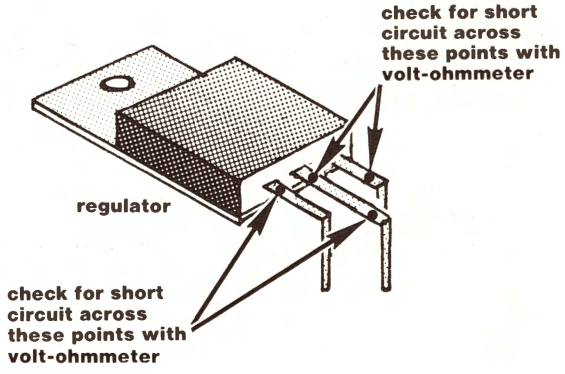


Figure 10.

MEMORY ADDRESS ASSIGNMENT

This board is addressable as a single 16K block on any K boundary. DIP switch positions 1-4 select which 16K a of memory the board will be associated with as follows:

1 - 0000 - 3FFF (hex)
2 - 4000 - 7FFF (hex)
3 - 8000 - BFFF (hex)
4 - C000 - FFFF (hex)

Only one of these four switches may be up (on) at a time, however, all four may be down which will deselect the board (content of the memory may neither be changed nor read) and the board is transparent to the system allowing other boards to be addressed to that area as long as it is disabled.

MEMORY PROTECT SWITCHES

DIP switch positions 5-8 are write enable switches. Inversely they may be used for manual write protection of the memory. Each enable protects 1/4 of the 16K block of memory (4K) as follows:

5 - 1st 4K of 16K block
6 - 2nd 4K of 16K block
7 - 3rd 4K of 16K block
8 - 4th 4K of 16K block

Any combination of these four switches may be up/on with up write enabling the particular 4K area and down write protecting that area.

WRITE STROBE SELECT SWITCH

This switch selects use or non-use of the S-100 buss signal MWRITE (pin 68). This signal is normally produced by a front panel and would not be available in front panel-less machines; however, some CPU boards now implement this signal. If MWRITE is present in your machine, place the slide switch in the center position. If not present, this switch should be in the up position.

PHANTOM LINE

In response to increasing numbers of users who have requested inclusion of "Phantom" line, Buss pin 67 which is often used for implementing power on jump features. This board is designed for use with active or inactive phantom lines. Boards which are revision "A" or earlier have this line solidly connected. Revision "B" and later boards require a jumper to implement this feature (see figure 11).

CAUTION

Some manufacturers use the PHANTOM line (buss pin 67) for a refresh signal. This will conflict with the PHANTOM feature, and cause boards with PHANTOM to fail. If your system has this conflict it must be resolved by either eliminating the refresh signal on the CPU board or disabling the PHANTOM feature on this and other boards.

On revision "A" or earlier boards the Phantom feature may be disabled by cutting the trace where it leaves buss pin 67. On revision "B" and later boards the PHANTOM feature is disabled if the jumper is NOT CONNECTED on the trace to buss pin 67 (see figure 11).

If you want the PHANTOM feature, the conflicting refresh signal may be eliminated (IF NOT USED ELSEWHERE IN THE SYSTEM) by cutting the trace connected to buss pin 67 on the CPU board. BE SURE OF YOUR SYSTEM CONFIGURATION BEFORE CUTTING ANY TRACES.

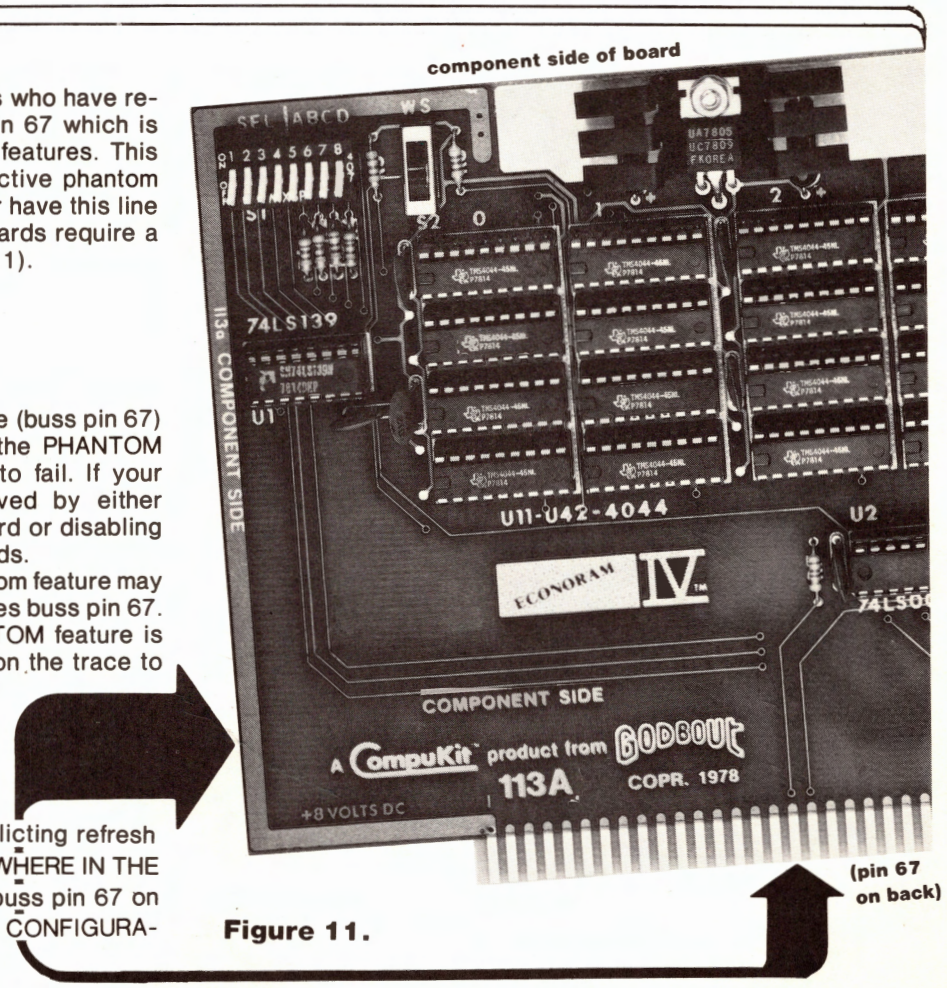
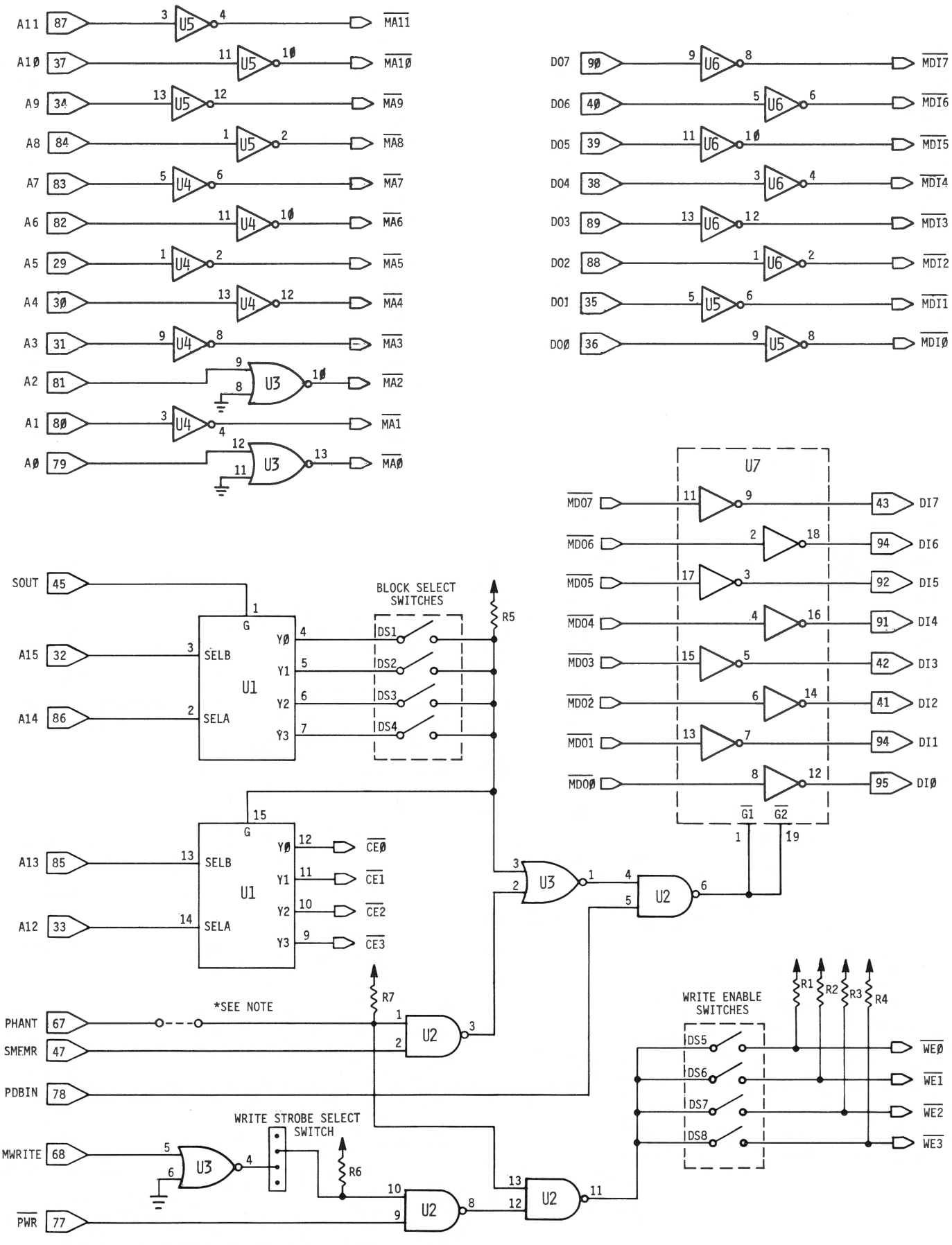
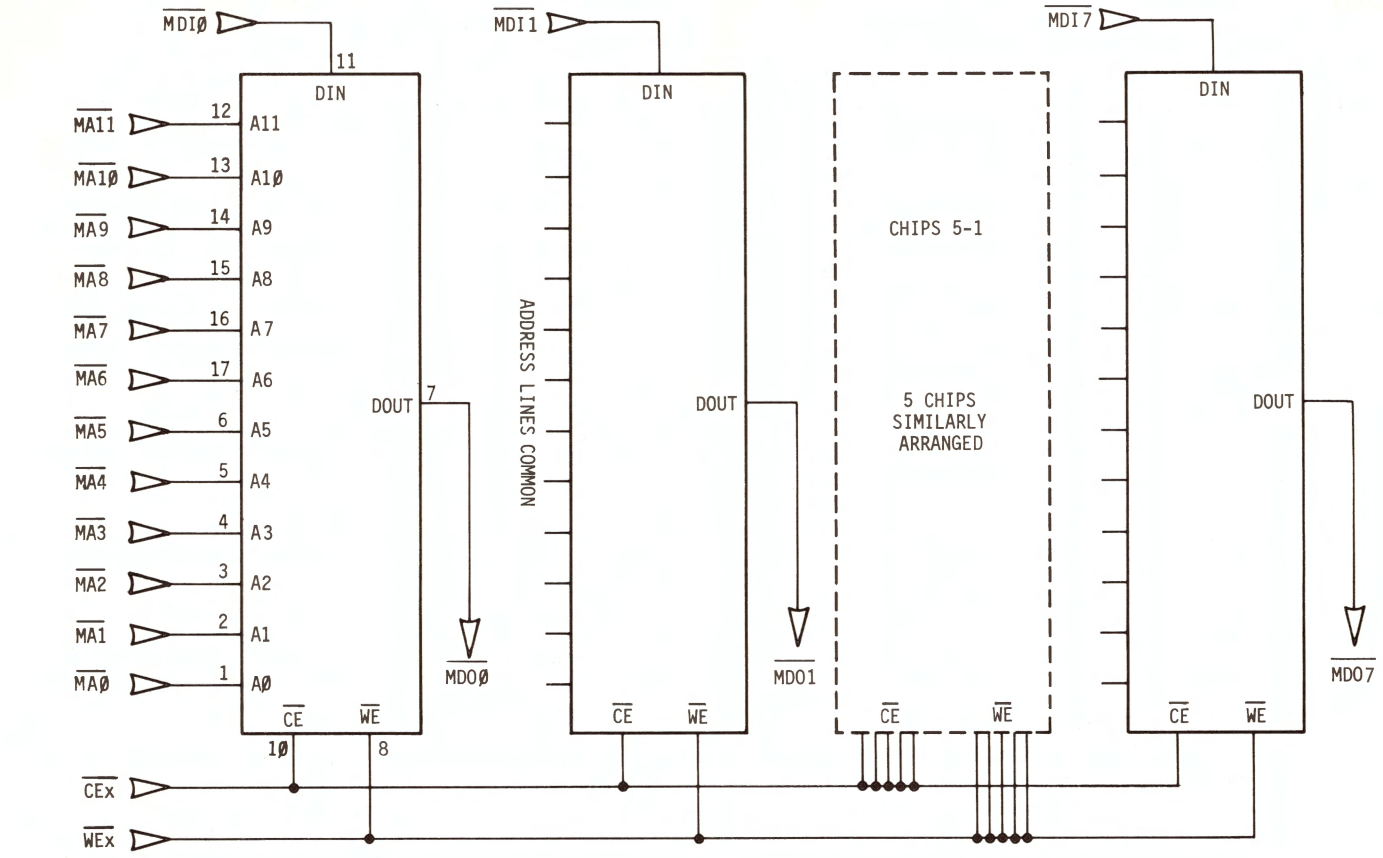


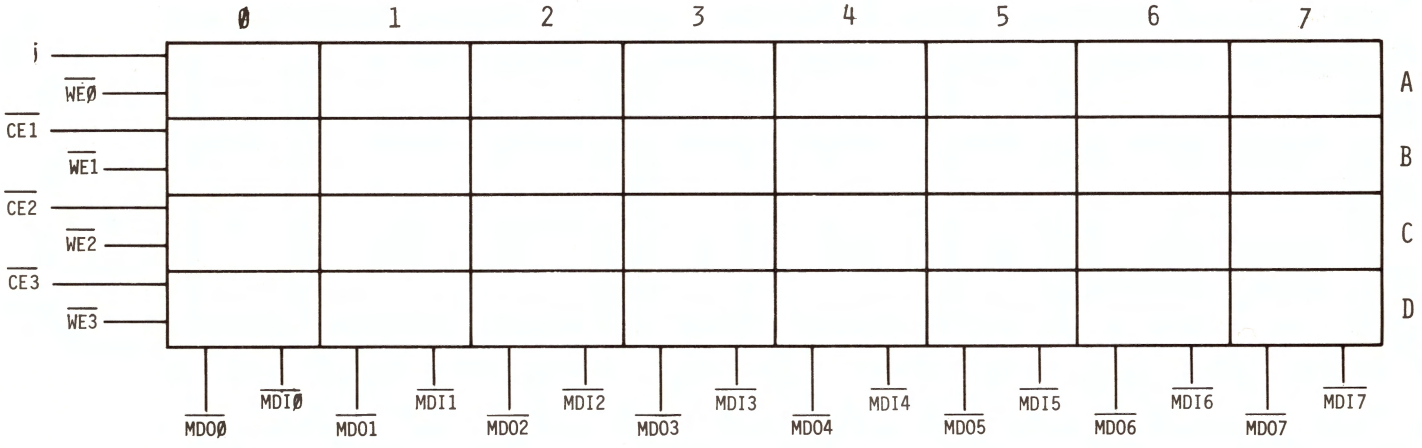
Figure 11.

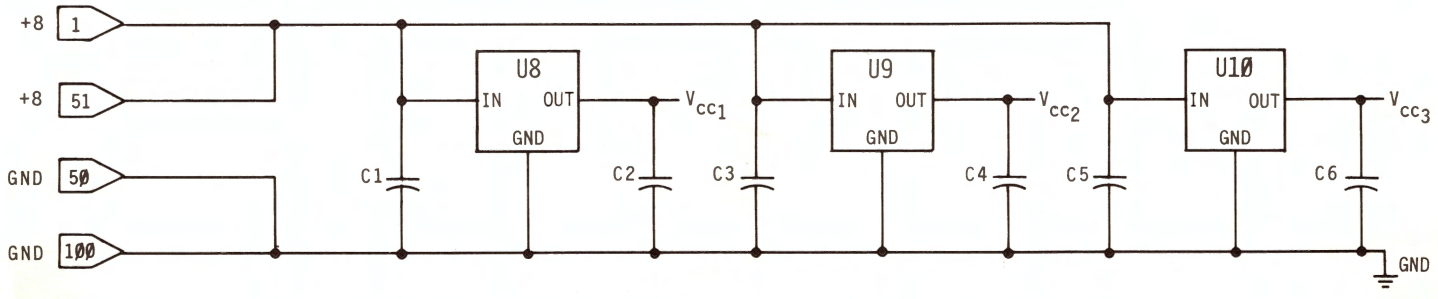
SCHEMATIC - ECONORAM IV



NOTE: On revision "A" boards this trace is solid; on revision "B" and later boards require a jumper to implement.







SCHEMATIC - ECONORAM IV

MEMORY TESTING

If the memory board seems to be working properly, the Memory Testing Routine (figure 13) can be used to give the board a more thorough workout. It is rather slow; but will do the job well. It can be entered via editor/assembler or front panel switches.

The routine is set up to test 16K from 4000 hex up to 8000 hex. This may be changed by entering a different starting address at "STRT" (3001 - 3002) and/or a different end address at "END" (3004 - high order byte only).

If the memory passes the test it starts over again. You may on the other hand, insert a jump instruction at "MARK" to some user routine or, if desired the user may enter an output instruction or, can do a notification routine at "MARK" to show successful completion and restart.

If the memory fails the test, critical information is stored and the routine enters a software "HALT", that is a "jump to here' at "SHLT". Front panel lights, if any, will show this state. The user may then use the front panel or dump routines to display the following stored failure info:

3069*	"FDE"	= D, E pair . . . D is the fill character and E is the test character
306B*	"FHL"	= H, L pair . . . the failure address
306D*	"FOUT"	= the data expected at this address
306E*	"FIN"	= the data read from, the address

* address from *Memory Testing Routine Listing.*

The user may replace the "jump" at "SHLT" with a jump to a display or notification routine.

The difference between "FOUT" and "FIN" should indicate which bit is failing, indicating which chip or area is causing the problem.

This test will find most of the harder to distinguish errors.

MEMORY TESTING ROUTINE

```
3000 21 00 40          0010 STRT LXI  H,4001
3003 3E 80             0020 END  MVI  A,80H
3005 32 6E 30          0030      STA  FIN
3008 3E 10             0040      MVI  A,10H
300A 84                0050      ADD  H
300B 4F                0060      MOV  C,A
300C 16 00             0070      MVI  D,0
300E 1E FF             0080      MVI  E,0FFH
3010 22 65 30          0090 DONE SHLD STAD
3013 AF                0100      XRA  A
3014 47                0110      MOV  B,A
3015 7B                0120 SCND MOV  A,E
3016 5A                0130      MOV  E,D
3017 57                0140      MOV  D,A
3018 79                0150      MOV  A,C
3019 2A 65 30          0160      LHLD STAD
301C 72                0170 FILL MOV  M,D
301D 23                0180      INX  H
301E BC                0190      CMP  H
301F C2 1C 30          0200      JNZ  FILL
3022 2A 65 30          0210      LHLD STAD
3025 73                0220 NEXT MOV  M,E
3026 7B                0230      MOV  A,E
3027 BE                0240      CMP  M
3028 C2 6F 30          0250      JNZ  FAIL
302B 79                0260      MOV  A,C
302C 23                0270      INX  H
302D 94                0280      SUB  H
302E C2 4D 30          0290      JNZ  NDON
3031 B8                0300      CMP  B
3032 44                0310      MOV  B,H
3033 CA 15 30          0320      JZ   SCND
3036 3A 66 30          0325      LDA  STAD+1
3039 00                0330 MARK NOP
303A 00                0331      NOP
303B 00                0332      NOP
303C 3A 6E 30          0340      LDA  FIN
303F B9                0350      CMP  C
3040 CA 00 30          0360      JZ   STRT
3043 79                0370      MOV  A,C
3044 67                0380      MOV  H,A
3045 2E 00             0390      MVI  L,0
3047 C6 10             0400      ADI  10H
3049 4F                0410      MOV  C,A
304A C3 10 30          0420      JMP   DONE
304D 22 67 30          0430 NDON SHLD NXAD
3050 7A                0440 LOPB MOV  A,D
3051 BE                0450 LOPA CMP  M
3052 C2 6F 30          0460      JNZ  FAIL
3055 2C                0470      INR  L
3056 C2 51 30          0480      JNZ  LOPA
3059 79                0490      MOV  A,C
305A 24                0500      INR  H
305B BC                0510      CMP  H
305C C2 50 30          0520      JNZ  LOPB
305F 2A 67 30          0530      LHLD NXAD
3062 C3 25 30          0540      JMP  NEXT
3065                   0550 STAD DS   2
3067                   0560 NXAD DS   2
3069                   0570 FDE  DS   2
306B                   0580 FHL  DS   2
306D                   0590 FOUT DS   1
306E                   0600 FIN  DS   1
306F 22 6B 30          0610 FAIL SHLD FHL
3072 32 6D 30          0620      STA  FOUT
3075 7E                0630      MOV  A,M
3076 32 6E 30          0640      STA  FIN
3079 EB                0650      XCHG
307A 22 69 30          0660      SHLD FDE
307D C3 7D 30          0670 SHLT JMP SHLT
3080                   0680 *
```

CIRCUIT DESCRIPTON

The heart of Econoram IV is the TMS4044/MM5257 static memory IC (RAM), which can store 4096 single bits of information (thus, each is a "4K x 1" memory IC). Unlike standard RAMs, those included with your kit are specifically designated by the manufacturer as low power, high speed parts.

These ICs are arranged in rows that are 8 ICs wide. This way, each row can store 4K x 8 bits of information. Paralleling 4 of these rows together produces a total memory storage of 16K x 8 bits. (Note that the bit number corresponding to a given column of ICs is indicated along the top edge of the memory array).

Now that we have this storage, there are still other aspects we must consider. First, we need to address a specific location in memory; and, we need to be able to *write* data into the memory, or *read* data from the memory.

The schematics on pages 8 and 9 show the address circuitry along with the other Econoram IV circuitry. Each memory IC requires 12 address bits (A0-A11) to access any one of the 4096 bits available in the IC. These address bits are generated by the CPU and are buffered by a number of inverters. After buffering, a particular address is presented to all IC address selection pins. However, we additionally need to select which particular row of ICs is to react to the given address. This requires 4 more address bits (A12-A15), which are decoded and used to enable the desired row of ICs (note row markings along the right hand side of the memory array).

When data is to be written into memory, it first passes through 8 inverting buffers before being put on the data pins of the RAMs (buffering prevents loading of the data buss). Data to be read on to the data buss from memory passes through 8 TRI-STATE® inverting buss drivers; when data is not being read on to the buss, the outputs of these inverters are in a high-impedance or "disconnected" state.

An unfortunate fact of life is that logic ICs generate switching transients that travel along the power supply lines. If these transients work their way into the logic circuitry, problems can appear. To prevent such occurances, bypass capacitors are tied across the power lines at regular intervals in the memory array and at every support IC.

This board is guaranteed to operate at 4MHz over the full temperature range (0° - 70° C ambient) and to draw less than 2000 mA (2 amps). Our typical measured currents were less than 1600 mA at cold start-up, rapidly decreasing to around 1200-1400 mA, depending on the surrounding temperature. We have heard similar reports from the people already using these boards.

It is interesting to note that static RAM technology has progressed to the point that this high-performance static RAM board is comparable in cost and power consumption to dynamic memory boards.

THANK YOU

This board is the result of much time, work and experience on the part of a number of people.

We strive for a board that doesn't just work the first time, but continues to give reliable operation for a long time. If we can be of any help to you in applying this board, or if you have any questions, please let us know. As always, we solicit your comments, letters, and new product suggestions.

HAPPY COMPUTING!

CUSTOMER SERVICE INFORMATION

Our paramount concern is that you be satisfied with any Godbout CompuKit product. If this product fails to operate properly, it may be returned to us, see warranty information below.

If you have any questions about assembly, performance, specifications or need further information feel free to write us at:

P.O. Box 2355, Oakland Airport, CA 94614.

When writing, please be as specific as possible concerning the nature of your query. We maintain a 24 hour a day phone, for taking orders, (415) 562-0636. If you have problems or questions which cannot be handled by mail, this number can be used to connect you with our technical people ONLY during normal business hours (10am-5pm Pacific Time). Unfortunately, we cannot return calls, or accept collect calls.

LIMITED WARRANTY INFORMATION

Godbout Electronics will repair or replace, at our option, any parts found to be defective in either materials or workmanship for a period of 1 year from date of invoice. Defective parts *must* be returned for replacement.

If a defective part or design error causes a Godbout Electronics product to operate improperly during the 1 year warranty period, we will service it free (original owner only) if delivered and shipped at owner's expense to Godbout Electronics. If improper operation is due to an error or errors on the part of the purchaser, there may be a repair charge. Purchaser will be notified if this charge exceeds $10.00.

We are not responsible for damage caused by use of solder intended for purposes other than electronic equipment construction, failure to follow printed instructions, misuse or abuse, unauthorized modifications, use of our products in applications other than those intended by Godbout Electronics, theft, fire, or accidents.

Return to purchaser of a fully functioning unit meeting all advertised specifications in effect as of date of purchase is considered to be complete fulfillment of all warranty obligations assumed by Godbout Electronics. This warranty covers only products marketed by Godbout Electronics and does not cover other equipment used in conjunction with said products. We are not responsible for incidental or consequential damages.

Prices and specifications are subject to change without notice, owing to the volatile nature and pricing structure of the electronics industry.

"Econoram" and "Econoram IV" are trademarks of W.J. Godbout.

TRI-STATE® is a trademark of National Semiconductor Corp.

Contents of this booklet ©1979 by Godbout Electronics. All rights reserved. We encourage quotation for the purposes of product review if source is credited.

PIO

PIO

FUNCTIONAL DESCRIPTION

The PIO board provides for up to four input and four output ports of eight bits each parallel input and parallel output. Each input and each output port has its own latch and both input and output latches are provided with hand-shaking logic for conventional eight bit parallel transfers.

Connection to the input or output ports is made through board edge connectors at the top of the board on .10 inch centers and the fingers will accept the 3M flat cable edge connectors as well as most other .1 inch center-to-center board edge connectors.

The handshake logic on any input or output port will generate an interrupt. The priority level of the interrupt is selectable. The address of the four ports is four sequential addresses, and this block of four addresses may be jumper-selected to be any block of four sequential addresses in the 256 I/O address space. The board may also be addressed with memory-mapped I/O, in which case normal memory read or write instructions are used to read or write data to the Input/Output ports. When using memory-mapped I/O, board addressing is done by selectable jumpers for the lower byte of address and the upper byte of address is hex FF or octal 377.

Provision is made for each of the four output ports to drive eight LED's for a total of 32 on-board LED's.

This feature can be used to provide program-controlled output for dedicated processor applications of the IMSAI 8080 in which case this PIO board would be plugged in where the front panel would normally be mounted and a special photographic mask made to put in front of it with the appropriate labels for the specific purpose the controller is to be used. The front panel can still be used during development by plugging it into an extender card in another slot.

The board is double-sided glass-epoxy-laminate G10-type and all holes are plated through to eliminate the need for any circuit jumpers. The power regulator is provided with a heat sink and has current limiting for protection in case of an overload. The I/O ports utilize the Intel 8212 8-bit latch.

The +5 and ground pins on the input or output port connectors can be used to provide 5 volt power at up to 200 or 300 milliamperes total from the full board. In addition, approximately 100 additional milliamperes of +5 volt power would be available for each 8212 input or output port which is not installed in the PIO board. For example, if four input ports were installed, but only two output ports were installed, the 5 volt power that could be drawn from the connectors would raise from 300 milliamperes to 500 milliamperes.

'IO 4-4 91-1200013 Kit or 91-0200013 Assembled
'arts List

This package contains:

QUANTITY	IMSAI PART #	Description
1	36-0740201	7402
1	36-0740202	74LS02
2	36-0740401	7404
2	36-0740402	74LS04
1	36-0742701	7427
2	36-0743002	74LS30
1	36-0780501	7805 Regulator
8	36-0821201	8212
12	32-2010010	.1uF Disk Capacitor
1	16-0100002	Heat Sink, Thermalloy 6106-B-14
3	23-0400001	16 Pin IC Header, SAE c#6170-16
32	35-3000001	Red LED, Dialco 521-9200
32	30-3220362	220 Ohm, ¼ Watt Resistor (red, red, brown)
41	30-4100362	1K ohm, ¼ Watt Resistor (brown, black, red)
1	92-0000013	PC Board, PIO 4 Rev. 2
3	23-0800001	16 Pin Solder Tail Socket
2	32-2233070	33-25 Tantalum Capacitor (or 22-25)
1	20-3302001	6-32x5/16" Phillips Pan Head Machine Screw
1	21-3120001	6-32 Hex Nut
1	21-3350001	#6 Internal Star Lockwasher
10'	15-0000001	Solder
1	81-0000024	Chapter, PIO 4 Rev. 2

This is a revised parts list and may differ from that in your manual. Please use this list for parts check-off.

3/29/77

AP 4-5-77

IMSAI 8080

PIO 4

Copyright 1977 IMSAI Manufacturing Corporation
14860 Wicks Boulevard
San Leandro, California
Made in the U. S. A.
All rights reserved worldwide.

June, 1977

FUNCTIONAL DESCRIPTION

The PIO 4 board provides for up to four input and four output ports of eight bits each parallel input and parallel output. Each input and each output port has its own latch and both input and output latches are provided with handshaking logic for conventional eight bit parallel transfers.

Connection to the input or output ports is made through board edge connectors at the top of the board on .10 inch centers and the fingers will accept the 3M flat cable edge connectors as well as most other .1 inch center-to-center board edge connectors.

The handshake logic on any input or output port will generate an interrupt. The priority level of the interrupt is selectable. The address of the four ports is four sequential addresses, and this block of four addresses may be jumper-selected to be any block of four sequential addresses in the 256 I/O address space. The board may also be addressed with memory-mapped I/O, in which case normal memory read or write instructions are used to read or write data to the Input/Output ports. When using memory-mapped I/O, board addressing is done by selectable jumpers for the lower byte of address and the upper byte of address is hex FF or octal 377.

Provision is made for each of the four output ports to drive eight LED's for a total of 32 on-board LED's.

This feature can be used to provide program-controlled output for dedicated processor applications of the IMSAI 8080 in which case this PIO board would be plugged in where the front panel would normally be mounted and a special photographic mask made to put in front of it with the appropriate labels for the specific purpose the controller is to be used. The front panel can still be used during development by plugging it into an extender card in another slot.

The board is double-sided glass-epoxy-laminate G10-type and all holes are plated through to eliminate the need for any circuit jumpers. The power regulator is provided with a heat sink and has current limiting for protection in case of an overload. The I/O ports utilize the Intel 8212 8-bit latch.

The +5 and ground pins on the input or output port connectors can be used to provide 5 volt power at up to 200 or 300 milliamperes total from the full board. In addition, approximately 100 additional milliamperes of +5 volt power would be available for each 8212 input or output port which is not installed in the PIO 4 board. For example, if four input ports were installed, but only two output ports were installed, the 5 volt power that could be drawn from the connectors would raise from 300 milliamperes to 500 milliamperes.

THEORY OF OPERATION

The board enable is the output of the 74LS30 in position C9. Input to this 8 input NAND gate is the true or complement address bits 2 through 6, according to how they are jumpered. The input and output status bits are logically ORed and the output or its complement is also jumpered to the NAND gate in position C9. These two are used for I/O reference instructions or these two inputs to the NAND gate are taken from the complement of the status input or output instruction and the high address line which comes from the 74LS30 in position C6. This NAND gate in position C6 is active when all the high order of address bits 8 through 15 are true, that is, high. Address 0 and 1 and their complements are fed into a one-of-4 decoder consisting of the 7427 in position and part of the 7402 in position C11 along with one inverter.

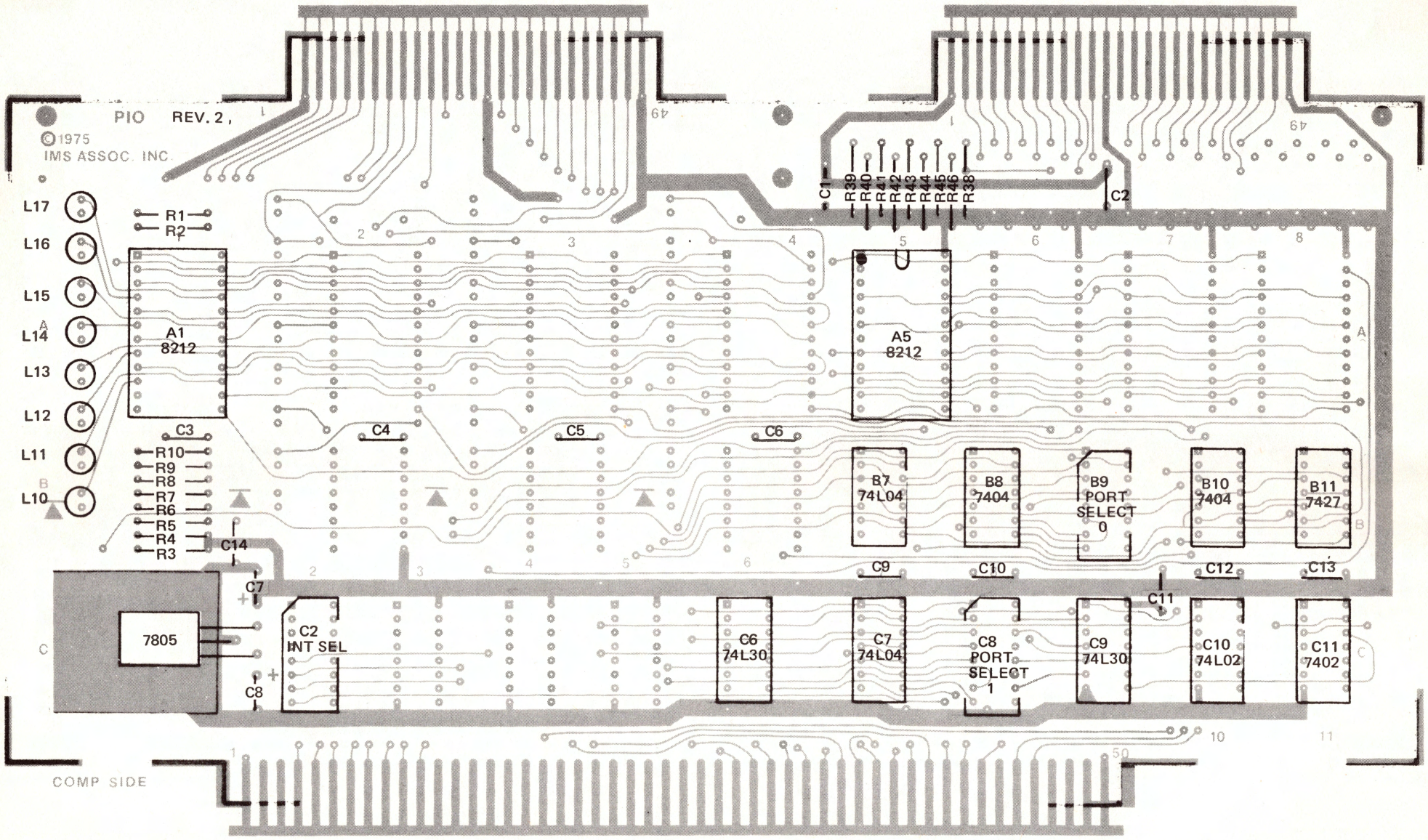
Also as a condition in this one-of-four decoder is the board enable. The outputs of this one-of-four decoder are fed directly to the enable pins on the respective 8212 input or output ports. The DATAIN bus on the IMSAI 8080 system is driven directly from the output of the four input latches. This is a tri-state output and is enabled only when the chip is selected by the one-of-four decoder.

The DATA OUTPUT bus in the IMSAI 8080 goes directly to the four 8212 output ports. The second enable line on each of the input ports is connected to the PROCESSOR DATA BUS-IN signal such that the data is placed on the IMSAI 8080 bus during the time that the processor wishes to read it. The other device select line in output port 8212's is driven by the ORed condition of the PROCESSOR WRITE STROBE or FRONT PANEL WRITE STROBE, these coming from pins 77 and 68 on the IMSAI 8080 back plane respectively. The PROCESSOR DATA BUS-IN signal appears on pin 78 of the IMSAI 8080 back plane.

Handling the interrupt levels from the four input and four output ports requires only the interrupt select jumper socket in position 2 so that the appropriate interrupt levels which are already originated by the 8212 chips can be connected as desired to the proper priority interrupt line on the IMSAI 8080 back plane. The remainder of the interrupt function is affected by the PIC-8 board, the Priority Interrupt/Clock board.

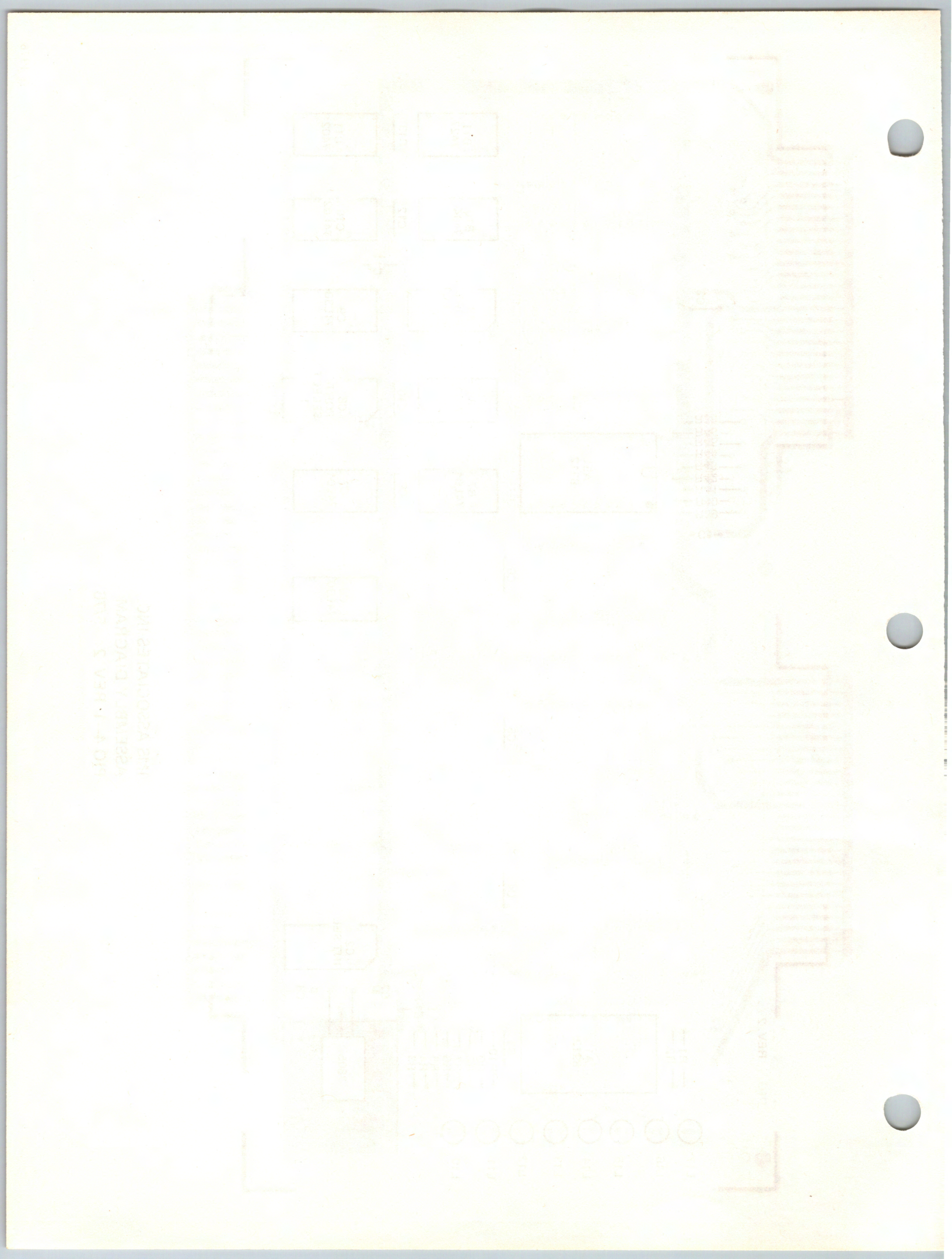
The LED's on the output ports are driven through the current-limiting resistor to +5 volts, so that when the output bit is low the LED is on. This orientation was chosen because the 8212's have a greater ability to sink current than they do to source current.

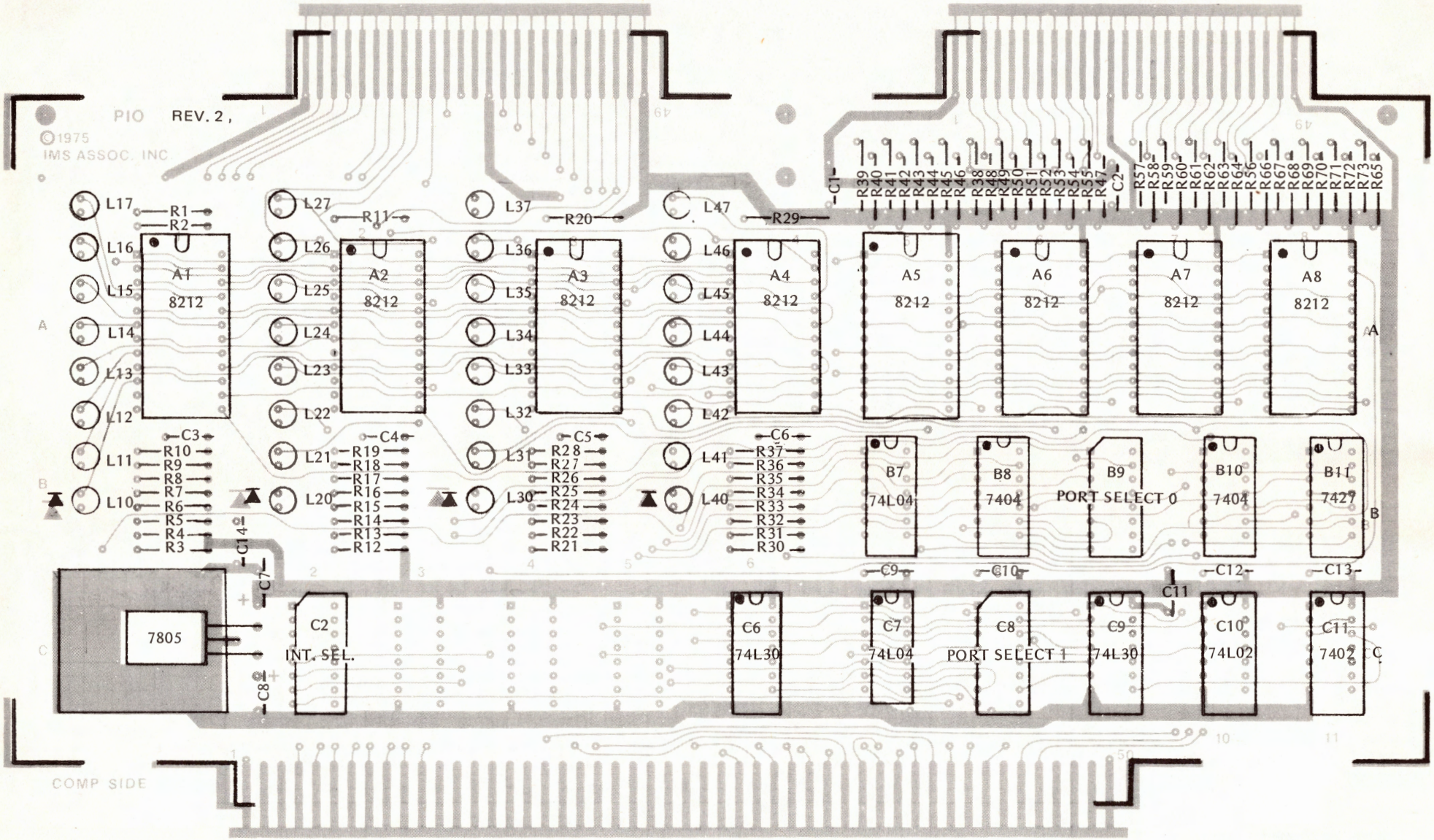
The strobe line into each 8212 input or output port is tied through a 1K resistor to +5 volts so that if the strobe line is not used, it will remain high and noise will not trigger signals on the input or output ports or the interrupt lines. All of the input lines have a 1K resistor to +5 volts, so that when the lines are not connected they exist in a defined state.



IMS ASSOCIATES INC.
ASSEMBLY DIAGRAM
PIO 4–1 REV. 2 5/76

©1976





IMS ASSOCIATES INC.

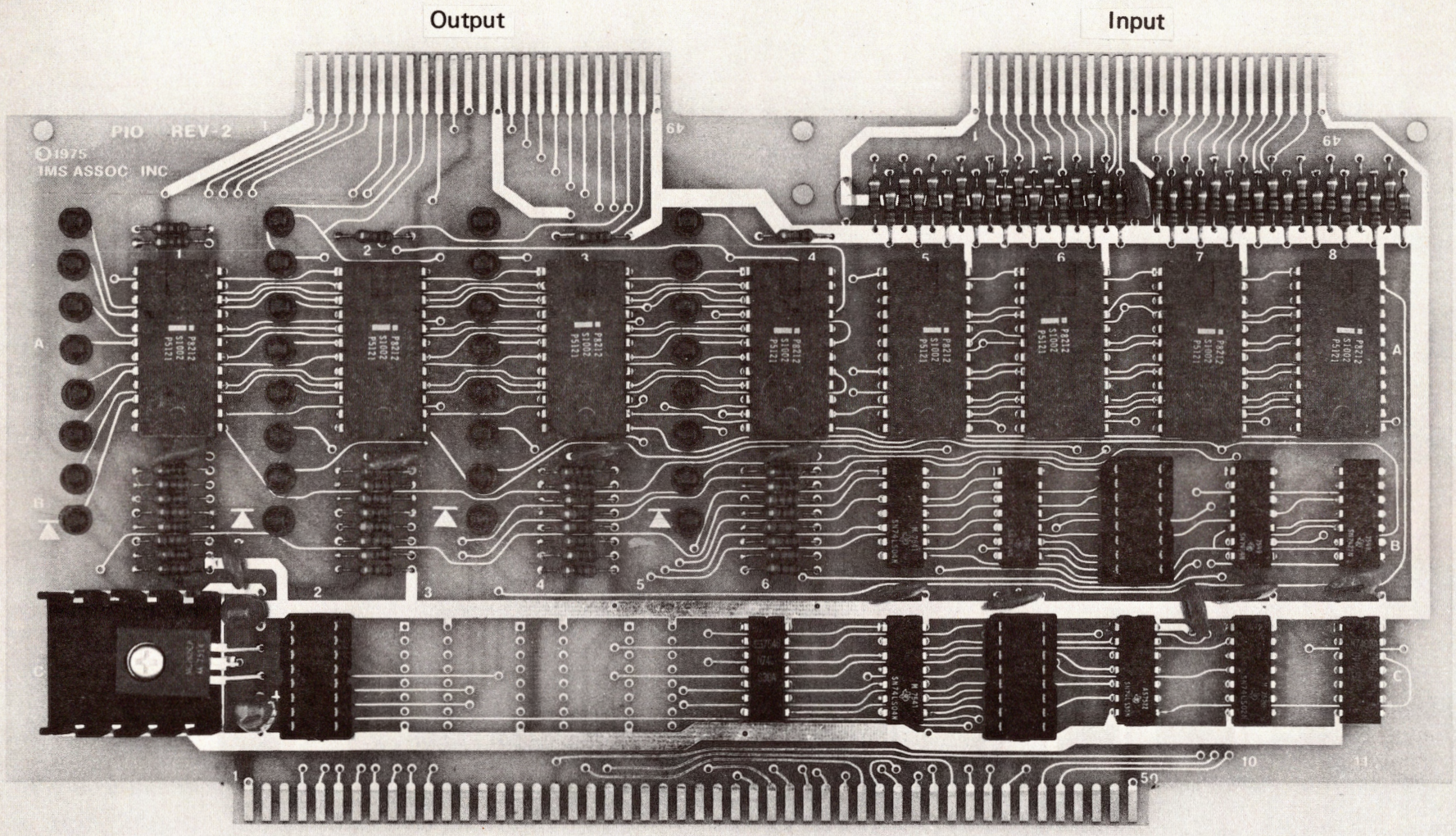
ASSEMBLY DIAGRAM

PIO 4–4 REV 2 1/76

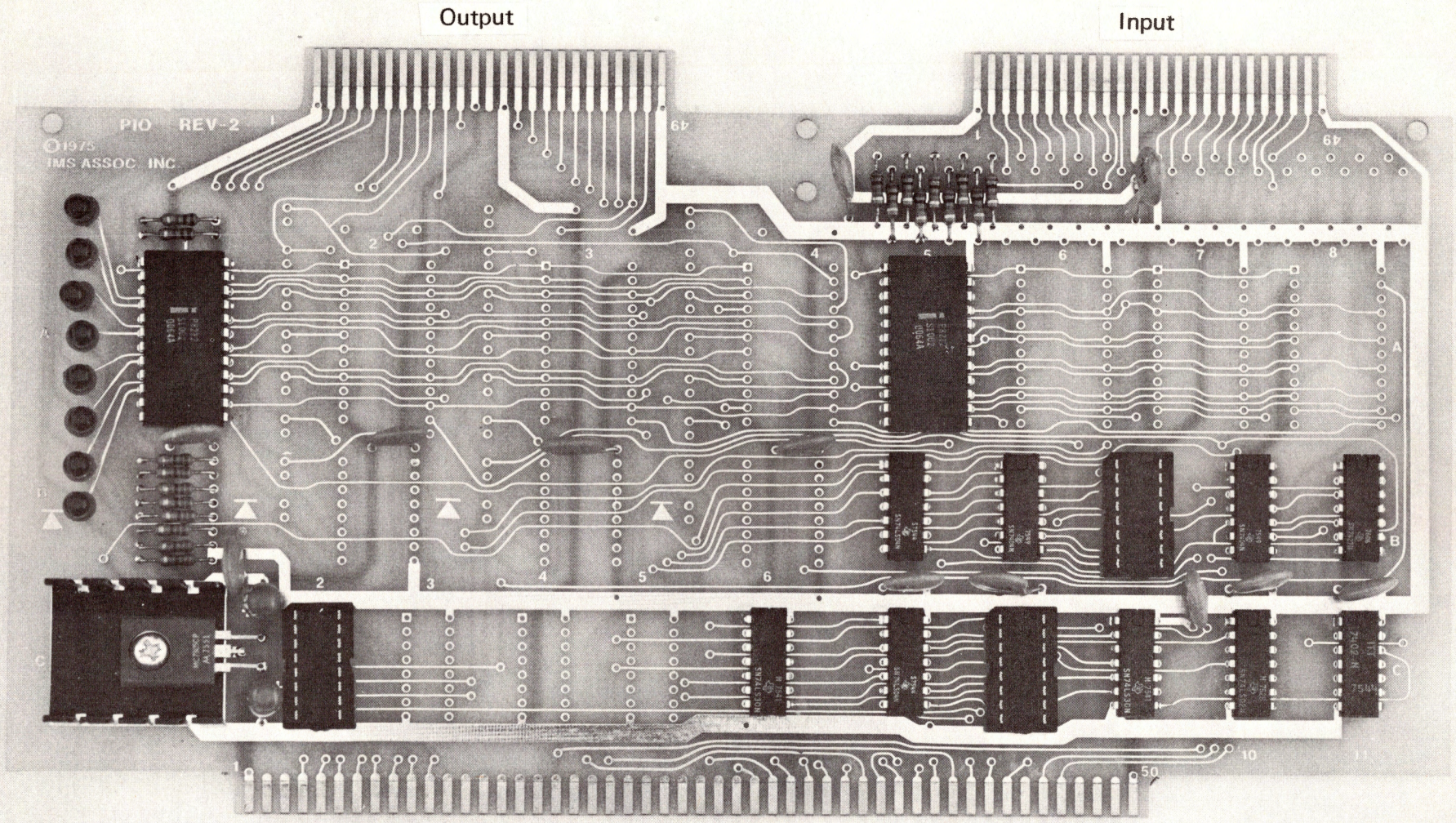
3/4/76

© 1977 IMSAI MFG. CORP.
SAN LEANDRO, CA.
ALL RIGHTS RESERVED WORLDWIDE
MADE IN U.S.A.

©1975



PIO 4 REV 2



PIO 4-1 REV 2

BOARD: PIO 4

ITEM	IMSAI PART #	QUANTITY	DESCRIPTION/IDENTIFYING MARKS
Solder	15-0000001	10'	
Heat Sink	16-0100002	1	Thermalloy Heat Sink/6106B-14
Screw	20-3303001	1	6-32x3/8" Phillips Pan Head Machine
Nut	21-3120001	1	6-32 Hex Nut
Lockwasher	21-3350001	1	#6 Internal Star Lockwasher
Header	23-0400001	3	16 Pin IC Header
Socket	23-0800001	3	16 Pin Solder Tail Socket
Resistor	30-3220362	8 32	(For PIO 4-1) 220 Ohm, ¼ Watt/ (For PIO 4-4) Red, red, brown
Resistor	30-4100362	41	1K Ohm, ¼ Watt/brown, black, red
Capacitor	32-2010010	12	.1uF Disk Ceramic
Capacitor	32-2233070	2	33-25 Tantalum
LED	35-3000001	8 32	(For PIO 4-1) Red Light Emitting Diode (For PIO 4-4)
7402	36-0740201	1	Quad 2 Input NOR/SN7402N
74LS02	36-0740202	1	Quad 2 Input NOR (Low Power Schottky)/ SN74LS02N
7404	36-0740401	2	Hex Inverter/SN7404N
74LS04	36-0740402	2	Hex Inverter (LPS)/SN74LS04N
7427	36-0742701	1	Triple 3 Input NOR/SN7427N
74LS30	36-0743002	2	8 Input NAND (LPS)/SN74LS30N
7805	36-0780501	1	5 Volt Positive Voltage Regulator/ MC7805CP
8212	36-0821201	2 8	(For PIO 4-1) 8 Bit I/O Port/D8212 (For PIO 4-4)
PC Board	92-0000013	1	PIO 4, Rev. 2

ASSEMBLY INSTRUCTIONS

( ) 1. Unpack your board and check all parts against the parts lists enclosed in the package.

( ) 2. If gold contacts on the edge connector appear to be corroded, use pencil eraser to remove any oxidation. NOTE: Do not use Scotchbright or any abrasive material as it will remove the gold plating.

RESISTOR INSTALLATION

( ) 3. Insert and solder the thirty-six 1K ohm, ¼ watt resistors (brown, black, red) at locations R38 through R73 in the upper right hand corner of the board as shown on the Assembly Diagram for PIO 4-4 and the 9 1K, ¼ watt resistors at locations R38 through R46 for PIO 4-1.

( ) 4. Insert and solder the other five 1K ohm, ¼ watt resistors at locations R1, R2, R11, R20 and R29 above the output port locations as shown on the Assembly Diagram for PIO 4-4; and the other two 1K ohm, ¼ watt resistors at locations R1 and R2 at output port location as shown on the PIO 4-1 Assembly Diagram.

IC INSTALLATION

NOTE: All Pin 1's are toward the upper left hand edge and away from the 100 pin connector.

( ) 5. Insert and solder the one 7402 at location C11.

( ) 6. Insert and solder the one 74LS02 at location C10.

( ) 7. Insert and solder the two 7404's at locations B8 and B10.

( ) 8. Insert and solder the two 74LS04's at locations B7 and C7.

( ) 9. Insert and solder the one 7426 at location B11.

( ) 10. Insert and solder the two 74LS20's at locations C6 and C9.

( ) 11. Insert and solder the eight 8212's at locations A1 through A8 for PIO 4-4; the two 8212's at locations A1 and A5 for PIO 4-1. NOTE: If fewer than

4 output or 4 input ports are being installed, they would normally be installed from left to right as the address of the output ports; for instance: Address 0 for position A3 and address 3 for position A4. The input port chips are arranged in a similar fashion from low to high address left to right, starting with position A5 and going to position A8. The choice of positions to be filled first can be made according to which addresses are desired on the input or output ports.

( ) 12. Insert and solder eleven .1uF Capacitors at locations C1 through C6, C9, C10 and C12 through C14 as shown on the Assembly Diagram. Be sure the capacitors are pulled in close to the board. NOTE: Although you have received 12 .1uF capacitors and the Assembly Diagram and photographs show one at location C11, this capacitor is not necessary for the proper functioning of your board and should not be installed.

In applications where the PIO board will be used in place of a front panel for programmed output in dedicated processor uses, care should be taken to lay the .1 disk ceramic bypass capacitors below the lowest LED position. This is to insure that nothing in the upper half of the board extends higher off the board than the LED's. If LED's are to be mounted, they can be mounted at this time.

( ) 13. Insert and solder the two 33uF tantalum capacitors at locations C7 and C8 as shown on the Assembly Diagram. NOTE: Observe polarity (+ to +) as shown on the board.

( ) 14. Insert and solder the 32 LED's at locations L10 through L17, L20 through L27, L30 through L37 and L40 through L47 as shown on the Assembly Diagram for PIO 4-4; the 7 LED's at locations L10 through L17 for PIO 4-1 as shown on the Assembly Diagram. Take a piece of cardboard, plastic or aluminum and cut it to approximately 3/8" wide for a strip about 3/4 inches long and then cut up the center. Don't cut all the way, so that the center saw cut can be placed on either side of the LED's leads, and the LED's can then be pushed into the holes and against the 1/8 inch spacer and soldered. The tool is then removed, and the LED should have a uniform spacing of 1/8 inch above the board. The 220 ohm or 230 ohm LED resistors should also be installed at this time if the LED's are installed. These are user supplied resistors and are not provided by IMSAI.

( ) 15. Insert and solder the 3 16 pin solder tail sockets at locations B9 and C8, to provide for addressing jumpers, and at location C2 to provide for priority interrupt jumpers.

REGULATOR AND HEAT SINK INSTALLATION

( ) 16. Before installing the regulator and heat sink, bend the 7805 regulator leads at 90 degree angles to facilitate mounting on the heat sink.

( ) 17. Insert the #6 screw through the regulator and heat sink on the component side of the board and attach through the lockwasher and nut on the circuit side. Tighten the screw carefully to insure proper alignment of the heat sink. Solder the 7805 leads.

USER GUIDE

The PIO 4 Board has four input ports and four output ports. Each port has an eight bit latch associated with it. These ports may be addressed in one of two different ways: First, addressed as an input/output port with input or output instructions; second, they may be addressed with memory reference instructions. The type of addressing is selectable by jumpers and the board cannot have both types of addressing at the same time. The four input ports form a block of addresses that are four sequential addresses and the four output ports form a block of four sequential addresses which are the same four addresses as the input port. In other words, the same address used with an input instruction to input on port number 0 is the same address used to output on port number 0.

When the board is being used with memory-mapped I/O, any 8080 instruction which either reads or writes a byte from memory can be used to either read or write respectively a byte from an input or output port on the I/O board. That is, a load accumulator, from the address that this board is jumper-selected to respond to, will load the accumulator with the data from the input port addressed. Each of the four input and each of the four output latches are equipped with data strobe lines. Each port has both an interrupt line and a strobe line which can be used as hand-shake signals for conventional parallel data transfers. In the case of the output ports, a low pulse on the strobe line will set the interrupt line low. The interrupt line changes on the falling edge of the strobe line and the strobe line would normally be kept high.

The interrupt line is made high again upon the trailing edge of the WRITE strobe of the processor which is writing a new eight bits of data into the output port. Thus, the strobe line would be the input hand-shaking line and the interrupt line would be the output hand-shaking line. The interrupt line may also be jumpered to one of the IMSAI 8080 priority interrupt lines on the back plane to effect an interrupt to the processor when it goes low, that is, when the strobe line has been pulsed low to indicate it has been taken by the peripheral device.

If it is not desired to use hand-shaking lines, it is not necessary to jumper them or take any other action. Successive bits may be put out to the output ports with no further action by any other device. In this case, the strobe line would remain high from the on-board pull-up resistor and

the interrupt line would remain high for lack of any strobe signal to affect it.

The input ports also have one strobe line and one interrupt line each. Each of the strobe lines for the input ports also has an on-board pull-up resistor. If the strobe line is not connected or if it is driven high, the data in the latch will follow the input lines. The program can read input from the input lines. The program can read input from the input lines and it will read the data that is present at the instant that the input instruction is executed. When the strobe line is made low the data that is present on the input lines at the falling edge of the strobe lines is latched into the input latch and remains there as long as the strobe line is held low. As soon as the strobe line is raised, the data in the latch will again follow the input lines. On the falling edge of the strobe lines the interrupt line will change from high to low.

This can be jumpered to the IMSAI 8080 priority interrupt lines to create an interrupt to the processor, and/or it may be used as an indication that the processor has not yet read the latched data. If, while the strobe line is being held low, the processor reads data from the input port, then the interrupt line will return high at the trailing edge of the read strobe, thus indicating to the peripheral device that the processor has read that data and the latch is available for latching the next data byte into it. Each input and each output port has its own strobe and interrupt line. They may be driven together or separately.

All four of the output port strobe, interrupt and data lines appear on the 50 pin connector on the upper left edge of the board, and all four of the input port strobe, interrupt and data lines appear on the 50 pin connector on the upper right hand edge of the board.

Also appearing on these connectors is ground and +5 volts. See the schematic diagram for pin numbers.

Both the input and the output connectors are designed to accept 3M-type flat cable board edge connectors. The flat cable may be run directly from the peripheral in through the flat cable clamp at the top back of the IMSAI 8080 Cabinet and directly to the edge connector which slips onto the top of the PIO 4 board.

Each of the data input lines on the input ports is tied to +5 volts through a 1K resistor so that unused lines will be read as high data level or true data level.

As an alternative, two EIA type connectors, 25 pins each, may be connected by way of flat cable and the 3M flat cable system to the board, so that the board connections can be taken to the EIA connectors with no hand wiring. This is true of the input port connector and the output port connector.

The connectors are arranged so that if the EIA 25 pin connectors are used, either two output ports or two input ports appear on the pins of the EIA connector along with both ground and +5 volts. If there are two devices, each needing only one output port, they may be connected without hand wiring both to the EIA 25 pin connectors by passing the flat cable through the first connector into a second so that two 25 pin connectors can be connected to the same flat cable; then only one port would be used in the first connector and the other port would be used in the other connector. This would permit separate plugs from separate peripheral devices without requiring any hand wiring of cables. The same, of course, is true for the input ports.

If desired for custom program display front panel-type use or just debugging use, the user can assemble the output ports with 8 LED's and current limiting resistors of either 220 or 330 ohms, so that the user has a visual indication fo the status of the output bits.

If the PIO 4 board is to be used in place of the front panel for custom program output, care should be taken during assembly to make sure that the disk ceramic bypass capacitors are not stood up but rather are laid over on top of the resistors they are next to so that they do not extend higher than the LED's.

The LED's on the output ports are turned on when the data bit is written as a 0. This was done because it was felt it was a more cost-effective solution for the user to put a complement instruction in his/her firmware than it was for IMSAI to put inverters in the hardware.

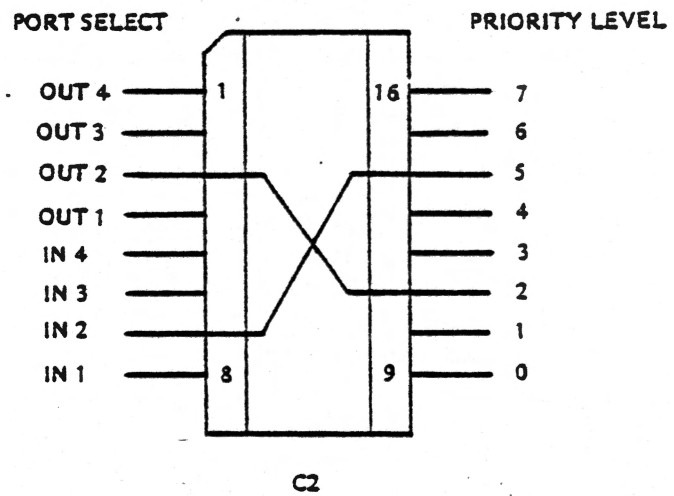
The +5 and ground pins on the input and output port connectors can be used to provide 5 volt power at up to 200 or 300 milliamperes total from the full board. In addition, approximately 100 additional milliamperes of +5 volt power would be available for each 8212 input or output port

which is not installed in the PIO 4 board. For example, if four input ports were installed, the 5 volt power that could be drawn from the connectors would raise from 300 to 500 milliamperes.

INTERRUPT SELECT JUMPER SOCKET

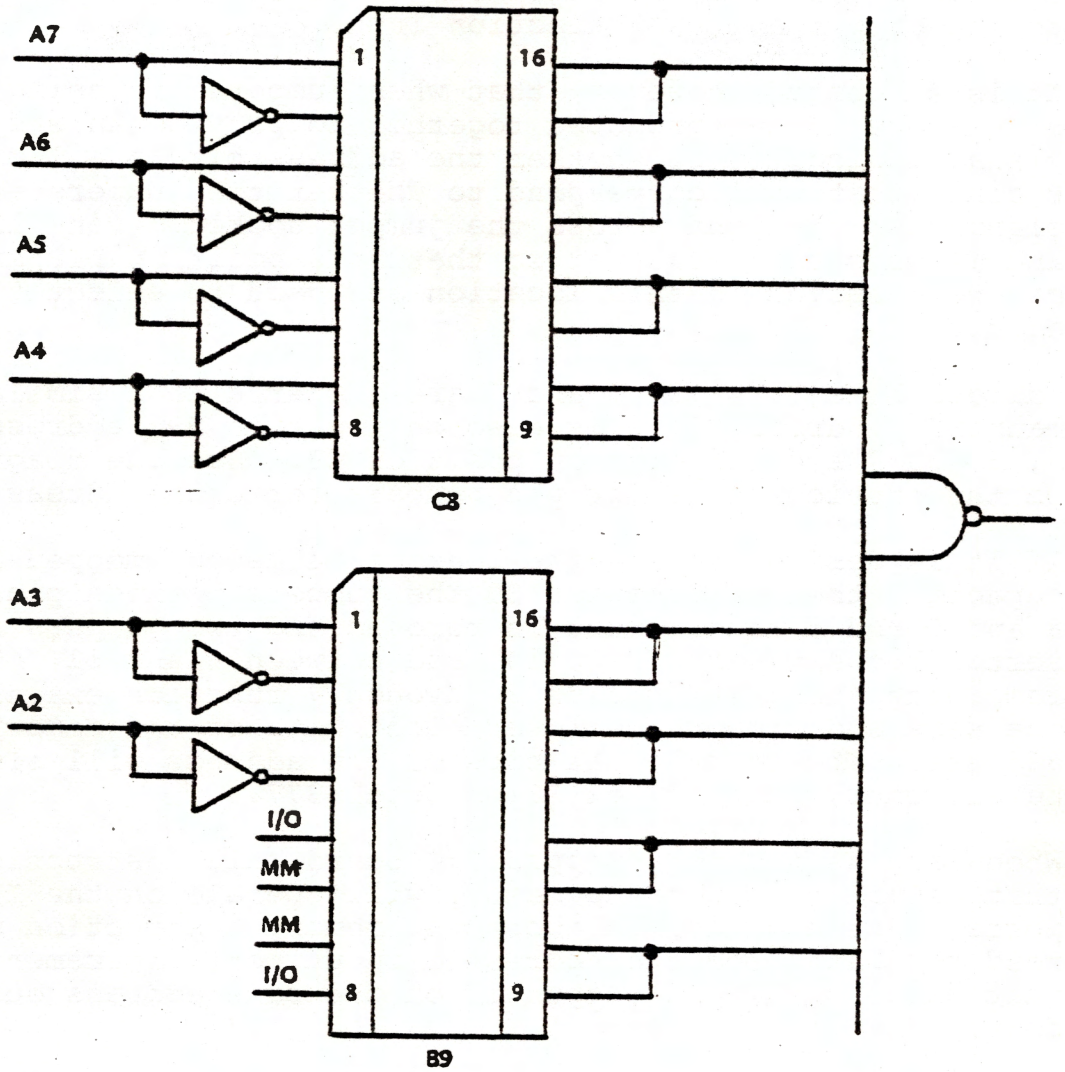
Position C2 on the PIO 4 board is the interrupt select jumper socket. Appearing at the pins of this socket are all eight of the priority interrupt lines for the IMSAI 8080, the four input interrupt lines and the four output interrupt lines of the PIO 4 board. Thus, any interrupt line desired to be used may be jumpered from the appropriate pin as noted on the diagram below.

If an interrupt is desired to be used, the jumper may be put between the interrupt line from the desired input or output port to the desired priority interrupt on the IMSAI 8080 back plane. The IMSAI PIC-8 board must be used to monitor these interrupt lines and originate the interrupt to the processor according to which line is requesting an interrupt. If more than one line is requesting an interrupt at the same time, the higher priority line rules. An example is shown below for connecting the interrupt line from input port 2 to level 5 priority and the interrupt line from output port 2 to level 2 priority interrupt.



C2

BOARD ADDRESS SELECTION JUMPER SOCKETS



The board address is selected by jumpers or a DIP switch in locations C8 and B9. There are two cases for which this board may be jumpered: 1) to respond to input/output instructions and 2) to respond to memory access instructions. The case of input/output instructions will be treated first.

In selection location B9, pins 8 and 9 must be jumpered together and pins 5 and 12 must be jumpered together. Address bits 0 and 1 determine which of the four input or output ports will be addressed. Port address bits 2 and 3

are also selected on location B9 with jumpers. If, for instance, address bit 2 is desired to be a 0 when the board responds, then pins 4 and 13 would be jumpered together. If address bit A2 was desired to be a 1, then either pins 3 or 13 may be jumperd together, since 13 and 14 are tied to the common address selection input.

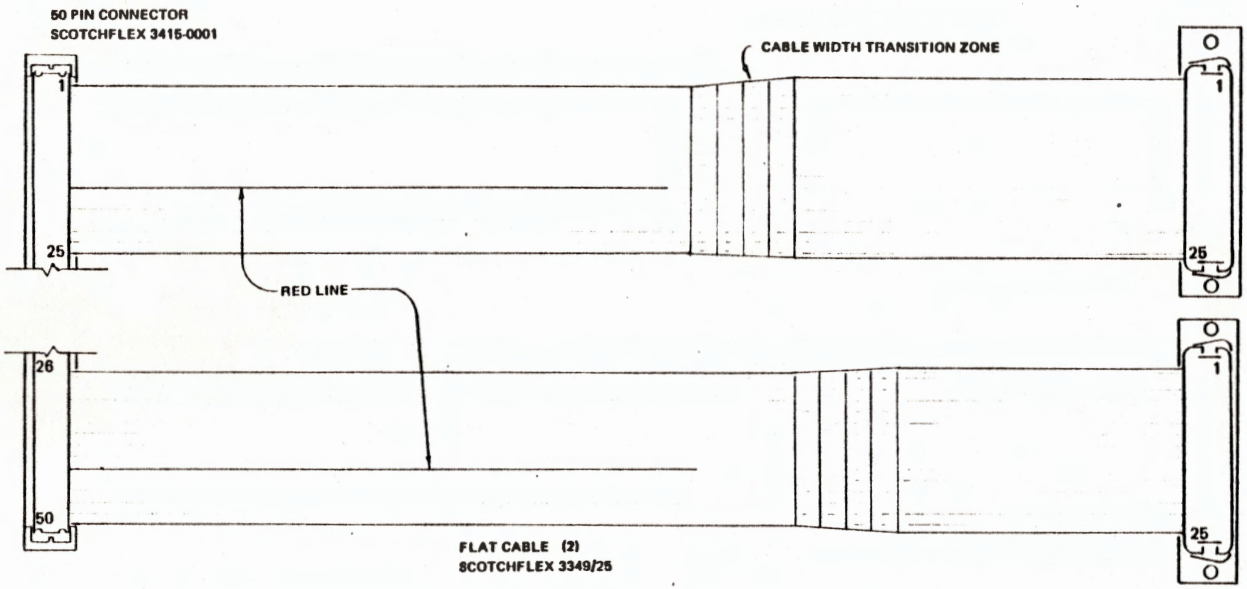
It is suggested, however, that when jumpers are being used, pins 3 and 13 be connected together to provide an easy visual indication of whether the address bit is a 1 or a 0 since that will correspond to whether the jumpers are slanted or straight across the jumper socket. Pins 13 and 14 were tied together so that an 8 position DIP switch can be inserted in this location and used to select the address.

Address bits 3, 4, 5, 6 and 7 are jumpered in a similar manner. Address bit 3 is also on location B9; address 4, 5, 6 and 7 are jumpered on position C8. See the diagram on the previous page for pin numbers for each address bit.

If it is desired to use the board in a memory-mapped I/O capacity, then in position B9 the jumpers between pins 8 and 9 and 5 and 12 must be removed and two jumpers inserted between pins 7 and 10 and between 6 and 11. The remaining jumpers for bits 2 through 7 function exactly the same and affect the lower eight bits of the memory address. The upper eight bits of the address will always be all ones, that is hex FF or octal 377.

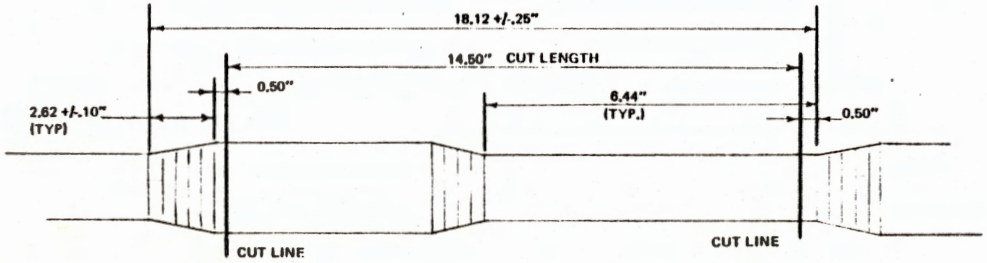
When used as a memory-mapped I/O board, all instructions that normally affect the memory will operate on the I/O ports. For example, an increment memory instruction would read the data from the addressed input port, increment that data by one and output it on the same address output port.

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED
0	ORIGINAL	8/76	
1	REVISED CABLE DIMENSIONS	8/76	



25 PIN FEMALE CONNECTOR
SCOTCHFLEX 3483-1000
HARDWARE SET
SCOTCHFLEX 3341-1

25 PIN FEMALE CONNECTOR
SCOTCHFLEX 3483-1000
HARDWARE SET
SCOTCHFLEX 3341-1



CABLE DETAIL

©1976
IMS ASSOCIATES INC.
SAN LEANDRO, CA.

TOLERANCES UNLESS OTHERWISE SPECIFIED FRACTIONS DEC. ANGLES ± ± ±		© 1977 IMSAI MFG. CORP. SAN LEANDRO, CA. ALL RIGHTS RESERVED WORLDWIDE MADE IN U.S.A.	
APPROVALS	DATE	IMSAI 8080 SYSTEM PIO CABLE B ASSEMBLY	
DRAWN YP	8/76		91-0400002
CHECKED A.L.A.	8/11/76	SCALE ——	SIZE B / DRAWING NO. 91-2040002
		DO NOT SCALE DRAWING	SHEET 1 OF 1

BISHOP GRAPHICS, INC.
REORDER NO. 20621

Base Address
10 Hex

A1 thru A8	8212
C7 B7	74LS04
B8 B10	7404
B9	JUMPER SOCKET
B11	7427
C2	JUMPER SOCKET
C6 C9	74LS30
C8	JUMPER SOCKET
C10	74LS02
C11	7402
C1 thru C6 C9 thru C13	.1mF
C7 C8	33mF
L10 thru L17 L20 thru L27 L30 thru L37 L40 thru L47	LED'S
R1 R2 R11 R20 R29 R38 thru R73	1K¼W
R3 thru R10 R12 thru R19 R21 thru R28 R30 thru R37	220 ¼W

IMS Associates Inc.
SCHEMATIC DIAGRAM
PIO REV. 2 2/76

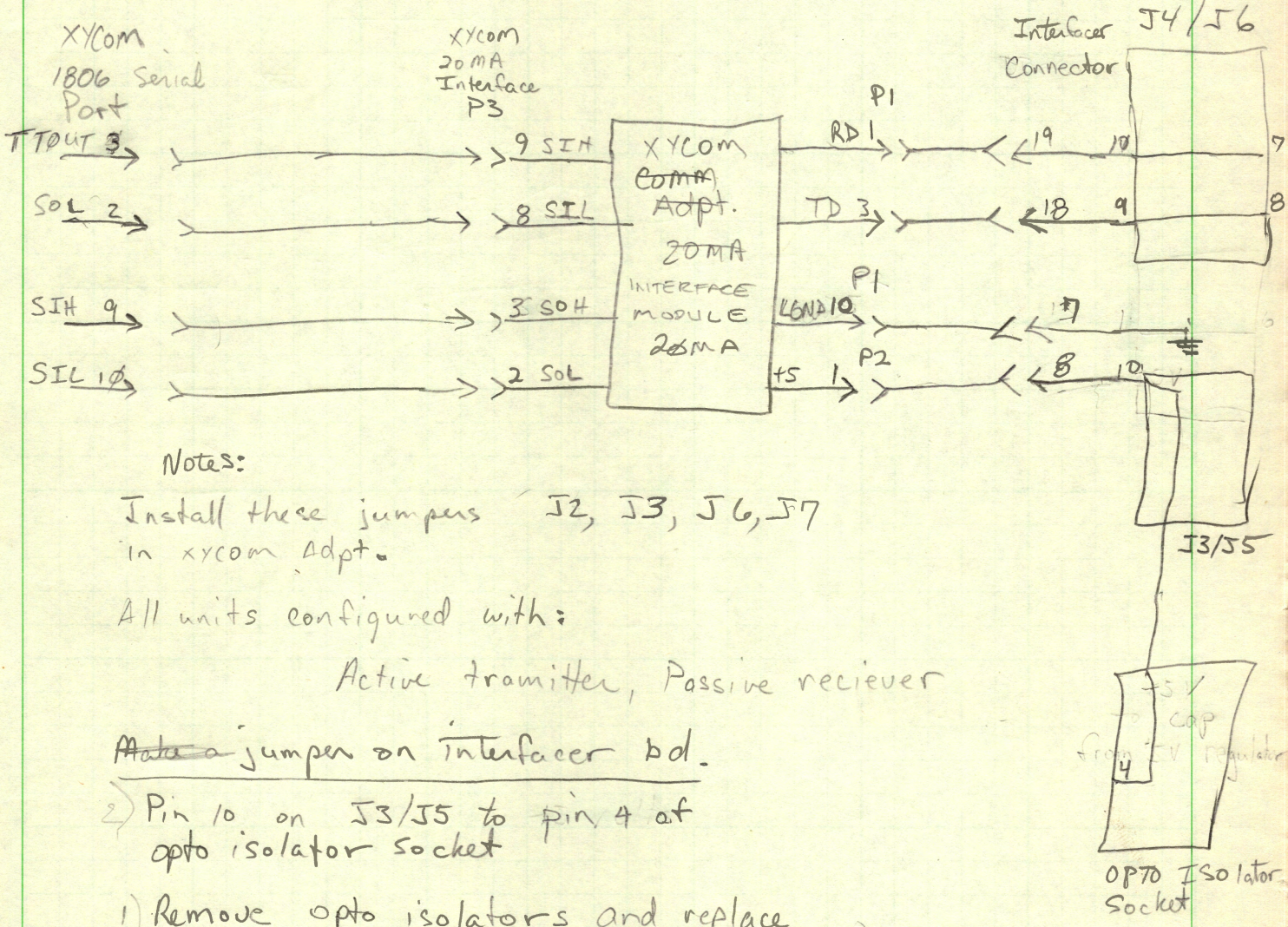
©1976

© 1977 IMSAI MFG. CORP.
SAN LEANDRO, CA.
ALL RIGHTS RESERVED WORLDWIDE
MADE IN U.S.A.

Misc. Interfacing

Interfacer Connector to XYCOM Serial port

Connection of XYCOM Communication module



Notes:

Install these jumpers J2, J3, J6, J7 in xycom Adpt.

All units configured with:

Active tramitter, Passive recievers

~~Make a~~ jumper on Interfacer bd.

2) Pin 10 on J3/J5 to pin 4 of opto isolator socket

1) Remove opto isolators and replace with 14 pin dip header

transmitter

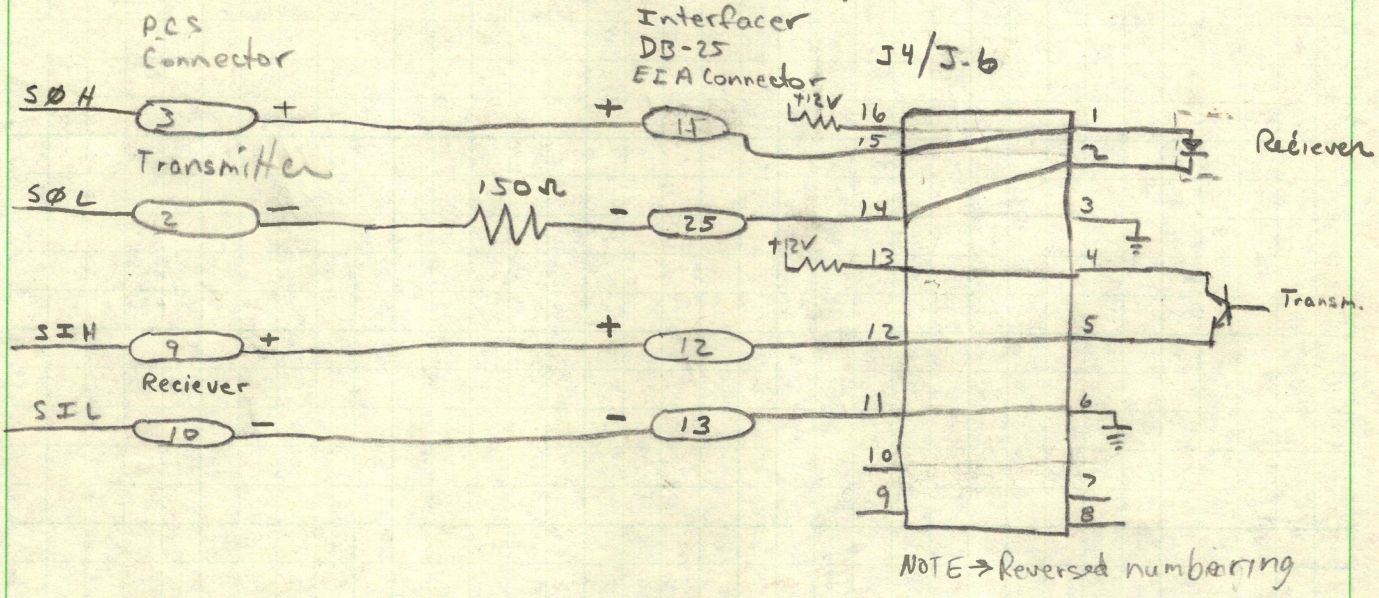
receiver

Interfacer
Current Loop Setup
for P.C.S. 1806 Computer

Scott Fahlman
Oct 30 1979

Note P.C.S Transmitter is active, passive reciever

Interfacer Transmitter is active, passive reciever

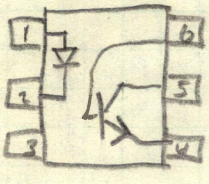


NOTE → Reversed numbering

No Connection should be made on headers J3/J5

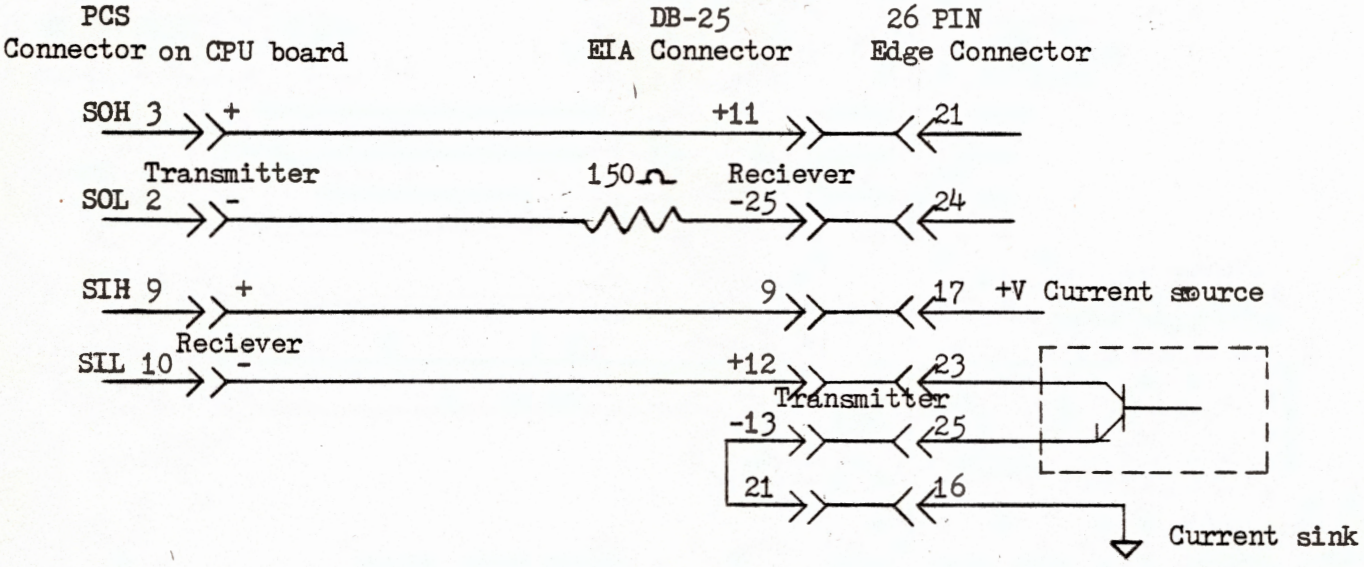
Note The 150Ω resistor limit the PCS tramitter Current to approx 20mA.

Interface will only go to 2400 baud unmodified



NATIONAL 42-381 50 SHEETS 5 SQUARE
42-382 100 SHEETS 5 SQUARE
42-389 200 SHEETS 5 SQUARE
Made in U.S.A.

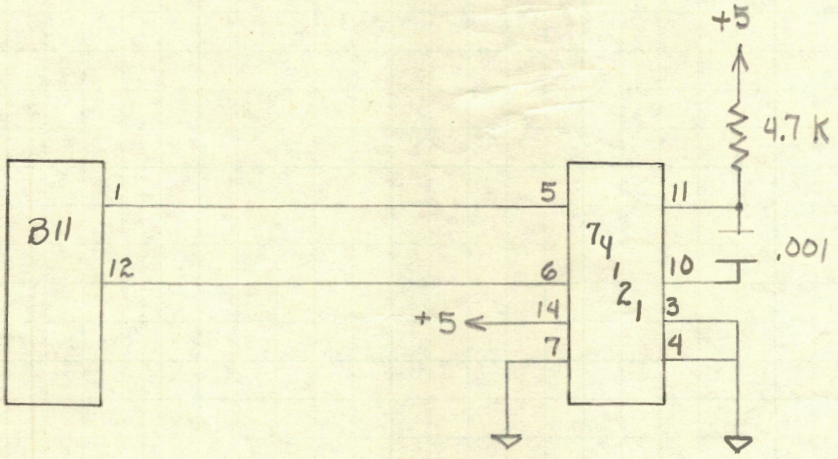
CONNECTION OF P.C.S. TO IMSIA SIO
20 MA LOOP



PCS uses an active transmitter and passive reciever.
Imsia uses an active transmitter and passive reciever.

Note: The 150Ω resistor limits the PCS transmitter current to approximately 20 MA.

S10 2-2 board 1/14/78 H.D.S.



To correct the duty cycle of the 9600A clock the above ckt. was connected to the I/O board by clip lead. The resulting duty cycle was 47%/53%

I_{Loop} 1/21/78 H.D.S.

ADM-3

Red - 17
Orn - 23
Brn - 24
Yel - 25

I_{MSAI}

Brn - 1
Orn - 3
Yel - 10
Red - 12

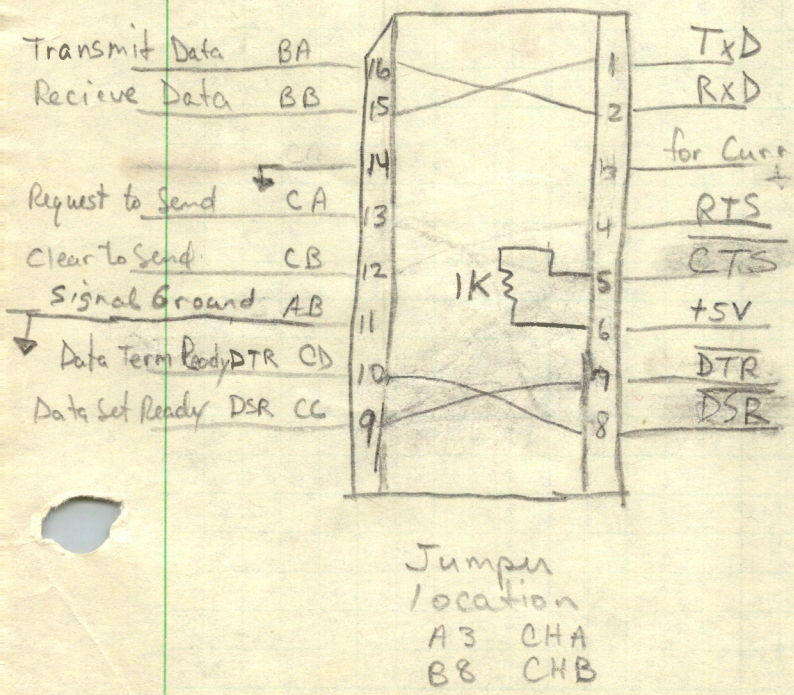
Printer Interface List — Oct 11 1978
T. I. 810
for Imsia SIO
Scott Rubin

EIA Pin	Signals Name	Source
2	Transmitted Data	Printer
3	Recieved Data	Computor
6	Data set Ready must be positive	Computor
7	Signal return	none
8	Carrier Detedmust be positive ?	Computor
4	Request to send - is held neg	Printer
11	Reverse Channel neg - busy pos. not busy	Printer
20	Data Terminal Ready pos - on line neg - off line and ready	Printer

DNR OPTION ENABLED

DSR - Data set ready pos
Carrier Detect pos

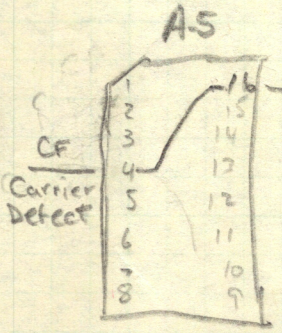
from Port



To 8251 UART
Note all signals are inverted before the 8251

Jumper location
A3 CHA
B8 CHB

Programing 8251



Set DTR low

Poll DSR

bit 7 → status port goes high when printer on line and not busy.

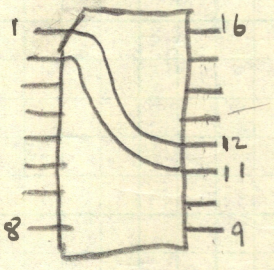
Scott Rubin June 22 1979

Current loop Notes for SIO Board

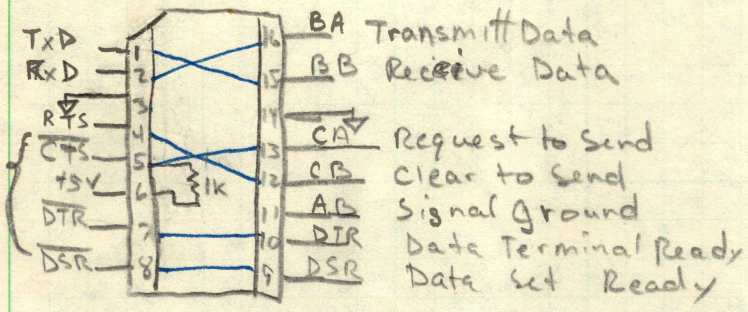
Baud Jumper location B-11

12	A common
11	B Common
1	9600 Baud Clock
2	4800 Baud Clock
3	2400 Baud Clock
4	1200 Baud Clock
5	600 Baud Clock
6	300 Baud Clock
7	150 Baud Clock
8	75 Baud Clock

sampk Channel A 9600 Baud
B 4800 Baud



Current loop Jumper location A3 Channel A
B7 Channel B



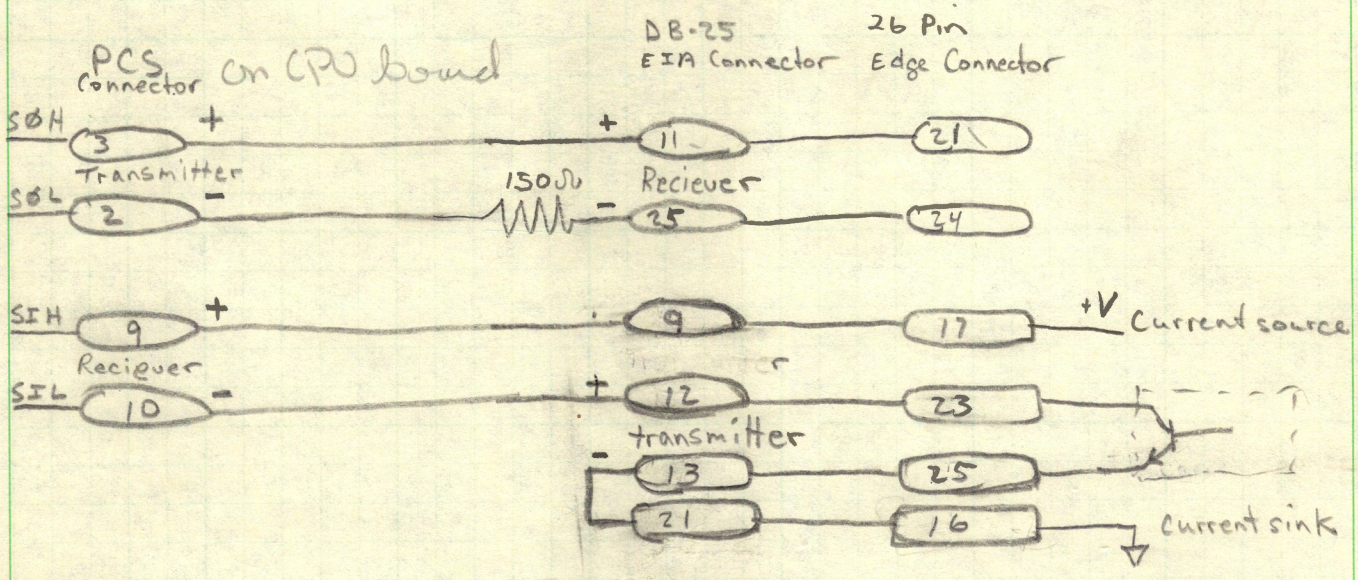
These signal from 8251 chip

Blue lines are for RS-232
Red lines for current loop

Ω

Scott Rubin Jun 22 79

Connection of P.C.S. to Imsia SIO
20 mA loop



PCS uses an active transmitter and passive recieuer } receiver
Imsia uses an active transmitter and passive recieuer

Note: The 150Ω resistor limits the PCS transmitter current to approx 20 mA.

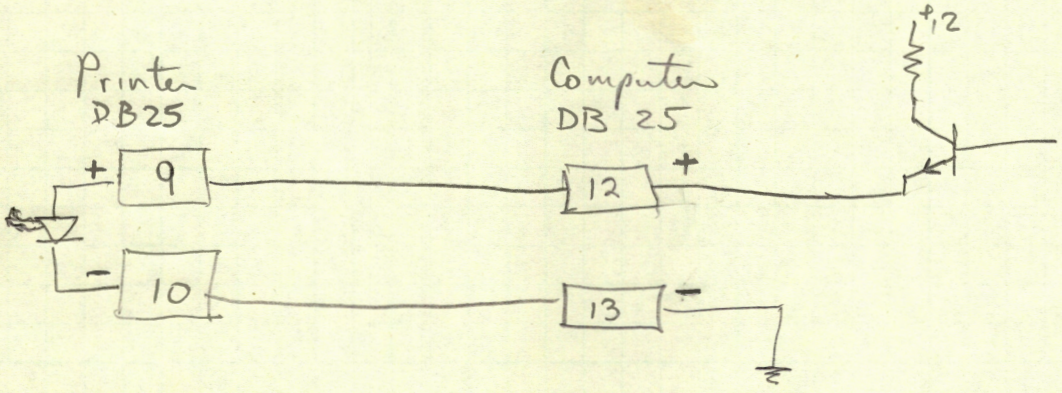
42-381 50 SHEETS 5 SQUARE
42-382 100 SHEETS 5 SQUARE
42-389 200 SHEETS 5 SQUARE
Made in U.S.A.
NATIONAL

C. Itoh | Printer To

Interfacer SIO
Cable

Printer C. ITOH Model 8300 R
Set up for a 20 MA Current loop

Computer - Godbout interfacer
Channel B set for 20 MA loop
Active transmitter



42-381 50 SHEETS 5 SQUARE
42-382 100 SHEETS 5 SQUARE
42-389 200 SHEETS 5 SQUARE
NATIONAL

SIO

SIO BOARD

FUNCTIONAL DESCRIPTION

The SIO Board provides a serial input/output capability for the IMSAI 8080 System. It contains two serial I/O ports, providing two complete RS232 full duplex data lines with all control signals. Data lines for both channels are provided in RS 232, TTL Level and current loop formats. Asynchronous or synchronous lines utilizing full or half duplex can be run with this board at any rate up to 9600 baud in the Asynchronous mode and 56,000 baud in the Synchronous mode.

The SIO Board may be jumper-selected to respond either to input and output instructions from the IMSAI 8080 System or to memory reference instructions for memory-mapped I/O.

Operation of the board requires 16 I/O port or address locations, which are selected by address bits 0 through 3. When the board is used with input and output instructions, address bits 4 through 7 form the remainder of the board address and are jumper selectable. When the board is used as memory-mapped I/O, the lower byte of address is jumper selected exactly the same as an I/O port address and the upper byte of address is hex FE or octal 376.

The SIO Board is structured around a pair of Intel 8251 USART (Universal Synchronous-Asynchronous Receiver-Transmitter) devices.

The 8251 chips provide for extensive program control of the input/output functions including the RS232 Control Line and sync character selection in the Synchronous mode and error condition sense and recovery. The board provides interrupt generation for received characters, empty transmitter buffers, and sync characters detected with provision for jumper selecting the priority of the interrupt. The interrupt works in conjunction with the Priority Interrupt/Clock board (PIC-8).

All functions may also be program controlled so that the full capability of the board is available to the machine without the use of interrupts. All RS232 level drivers and receivers necessary for two complete RS232 lines are included on the board.

Control lines included are DSR, DTR, RTS, CTS, and Carrier Detect. RS232 level drivers and receivers are also provided for receive and transmit clocks for use in Synchronous Mode. Jumper options permit the SIO board to be used either as the receiving (terminal) end of an RS232 line, or as the originating (computer) end.

Jumper options are available so that the two serial I/O ports may be used together so that the control lines are connected together on the two ports and the data lines are received and originated by the 8251 USARTS.

This configuration permits breaking an existing RS232 line and inserting the IMSAI 8080 System between the ends so that the control signals pass straight through and the IMSAI 8080 System intercepts, processes, and retransmits the data. This configuration is extremely useful where format adaptation or other changes must be made to data travelling on RS232 Systems.

Jumper-selectable baud rates are provided on the board for standard asynchronous and synchronous rates up to 9600 baud asynchronous and up to 38,400 baud synchronous. Other rates may be obtained through the use of the SIOC board which contains a jumper-programmable divider which mounts directly onto the SIO Board.

The two output connectors on the top of the board are designed to use the 3M flat cable system to connect directly to 25 pin EIA connectors so that no hand wiring is required to either receive or originate an RS232 line.

TTL and current loop serial input and output are connected to unused pins on the input/output connector. TTL levels are available on the connector for DTR, DATAIN, and DATAOUT, to provide maximum flexibility and utility. A current source is available on the connector for use with current loops. Current loop driving is done through opto-isolators for complete isolation of current loop lines.

Integrated circuit power regulation is provided with high quality tantalum and disc ceramic by-pass capacitors. The board is made on G10-type, 1/16 inch laminate with contact fingers gold-plated over nickel for reliable contact and long life. The remainder of the circuitry is tin-plated for good appearance and reliable solder connections.

Plated through-holes eliminate the need for any circuit jumpers. All jumper options are provided in 16 pin dual in-line package patterns so that jumpers may be installed on headers plugged into IC sockets for convenient and quick changing.

SIOC BOARD

FUNCTIONAL DESCRIPTION

The IMSAI SIOC Board is a small optional board used with the Serial Interface (SIO Board). The SIOC provides user selection of any USART clock frequency from 15 Hz to 56 KHz.

The generated clock frequency is determined by a binary value set in two 16-pin jumper sockets. An additional jumper socket allows selection of either the SIOC or the standard SIO USART clocks to channels A and/or B.

Physically, the SIOC Board measures 5.2 X 2.2", and piggy-back mounts to a standard SIO Board. Mounting hardware and decoupling capacitors are provided with the SIOC Board.

IMSAI 8080

SIO 2

Copyright 1976 IMSAI Manufacturing Corporation
14860 Wicks Boulevard
San Leandro, California 94577
Made in the U. S. A.
All rights reserved worldwide.

SIO 2 Board

FUNCTIONAL DESCRIPTION

The SIO Board provides a serial input/output capability for the IMSAI 8080 System. It contains two serial I/O ports, providing two complete RS232 full duplex data lines with all control signals. Data lines for both channels are provided in RS 232, TTL Level and current loop formats. Asynchronous or synchronous lines utilizing full or half duplex can be run with this board at any rate up to 9600 baud in the Asynchronous mode and 56,000 baud in the Synchronous mode.

The SIO Board may be jumper-selected to respond either to input and output instructions from the IMSAI 8080 System or to memory reference instructions for memory-mapped I/O.

Operation of the board requires 16 I/O port or address locations, which are selected by address bits 0 through 3. When the board is used with input and output instructions, address bits 4 through 7 form the remainder of the board address and are jumper selectable. When the board is used as memory-mapped I/O, the lower byte of address is jumper selected exactly the same as an I/O port address and the upper byte of address is hex FE or octal 376.

The SIO Board is structured around a pair of Intel 8251 USART (Universal Synchronous-Asynchronous Receiver-Transmitter) devices.

The 8251 chips provide for extensive program control of the input/output functions including the RS232 Control Line and sync character selection in the Synchronous mode and error condition sense and recovery. The board provides interrupt generation for received characters, empty transmitter buffers, and sync characters detected with provision for jumper selecting the priority of the interrupt. The interrupt works in conjunction with the Priority Interrupt/Clock board (PIC-8).

All functions may also be program controlled so that the full capability of the board is available to the machine without the use of interrupts. All RS232 level drivers and receivers necessary for two complete RS232 lines are included on the board.

Control lines included are DSR, DTR, RTS, CTS, and Carrier Detect. RS232 level drivers and receivers are also provided for receive and transmit clocks for use in Synchronous Mode. Jumper options permit the SIO board to be used either as the receiving (terminal) end of an RS232 line, or as the originating (computer) end.

Jumper options are available so that the two serial I/O ports may be used together so that the control lines are connected together on the two ports and the data lines are received and originated by the 8251 USARTS.

This configuration permits breaking an existing RS232 line and inserting the IMSAI 8080 System between the ends so that the control signals pass straight through and the IMSAI 8080 System intercepts, processes, and retransmits the data. This configuration is extremely useful where format adaptation or other changes must be made to data travelling on RS232 Systems.

Jumper-selectable baud rates are provided on the board for standard asynchronous and synchronous rates up to 9600 baud asynchronous and up to 38,400 baud synchronous. Other rates may be obtained through the use of the SIOC board which contains a jumper-programmable divider which mounts directly onto the SIO Board.

The two output connectors on the top of the board are designed to use the 3M flat cable system to connect directly to 25 pin EIA connectors so that no hand wiring is required to either receive or originate an RS232 line.

TTL and current loop serial input and output are connected to unused pins on the input/output connector. TTL levels are available on the connector for DTR, DATAIN, and DATAOUT, to provide maximum flexibility and utility. A current source is available on the connector for use with current loops. Current loop driving is done through opto-isolators for complete isolation of current loop lines.

Integrated circuit power regulation is provided with high quality tantalum and disc ceramic by-pass capacitors. The board is made on G10-type, 1/16 inch laminate with contact fingers gold-plated over nickel for reliable contact and long life. The remainder of the circuitry is tin-plated for good appearance and reliable solder connections.

Plated through-holes eliminate the need for any circuit jumpers. All jumper options are provided in 16 pin dual in-line package patterns so that jumpers may be installed on headers plugged into IC sockets for convenient and quick changing.

SIO THEORY OF OPERATION

To enable the SIO board, it must be properly addressed. In the I/O port addressed mode, address bits A4 through A7 are jumpered to the 74LS30 (8 input NAND) in C8. The status bits SINP and SOUT are NORed, this intermediate value inverted, and applied (via jumper on D6) to another of the NAND inputs. Remaining NAND inputs in this mode are jumpered (via D6) to a +5 volt level. Thus, when the selected address appears on A4-A7, and the MPU sends a SINP or SOUT pulse, the NAND output goes low and the board is enabled. See schematic.

In the memory-mapped I/O mode, the jumpering in socket C7 still selects an address. The high-order address is interpreted in another 8 input NAND (D8), and hard-wired to respond to the hex value FE. The jumper in socket D6 should be wired to put the inverted output of D8 into an input of C8, and the NORed output of the status bits SINP and SOUT directly connected to the (C8) NAND's input.

The +5 volt tie line jumper in D6 should not be connected for memory-mapped I/O. In this mode, when the corrected high and low order bits are on A4 through A15, and the MPU does not send a SINP or SOUT pulse, the board is enabled. See Diagram.

The SIO board has a bi-directional data bus on the board which connects to the 8251 chips and to the input and output portion of the SIO board control port. The bi-directional bus is connected to the DATA IN and DATA OUT busses on the IMSAI 8080 back plane through 8216 bi-directional bus driver chips. The board enable signal selects these bi-directional bus driving chips and the processor's data bus in signal (DBIN) is used to determine the direction of driving of the bi-directional chips.

8T97's are used to gate the control port data on the bi-directional data bus on the board. They are enabled by the DBIN strobe from the processor and address bit 3.

The 4 output bits of the control port on the SIO board are latched into the 74177 which is clocked by a combination of board enable and address bit 3 and the write strobe either from the processor or from the front panel.

The 8251 chips are selected by address bits 1 and 2, respectively, with address bit 0 determining whether the chip is in control or data mode. The read and write strobes are supplied to complete the control, enabling the chip to read data or write data onto the bi-directional data bus on the board.

The four control lines desired for interrupt generation are ORed through 7425 and the resultant value supplied to an interrupt select jumper socket (D3). The 7425 OR gate may be disabled by two of the output port bits (IEA or IEB) when interrupts are not desired.

The two megacycle system clock phase II is divided to provide the standard baud rates for jumper selection to channel A and B. It is first divided by 13 through the use of a 7493 with external gating. This produces a rate extremely close to 16 times 9600 baud.

Further division of two are made by 7493's to provide most of the other standard baud rates. 110 baud for a standard teletype is achieved by a divide by 11 from the 2400 baud line which is then divided by 2 to create a symmetrical output and supplied to the jumper socket for 110 baud.

The phase II clock, +5 volts and ground are also supplied to the data rate select socket for use by the SIOC board which connects to the SIO board through the data rate select socket (B11) to provide a jumper-selectable baud rate generator for special rates.

The data and control outputs of the 8251 chips are driven or received through 1488 or 1489 TTL to RS232 level converters as appropriate to the functions. The TTL levels for data and control are driven through open-collector peripheral drivers and a 220 ohm pull-up to +5 volts. The current loop input and output are driven through opto-isolators and are designed to work adequately with either 20 or 60 miliampere current loops.

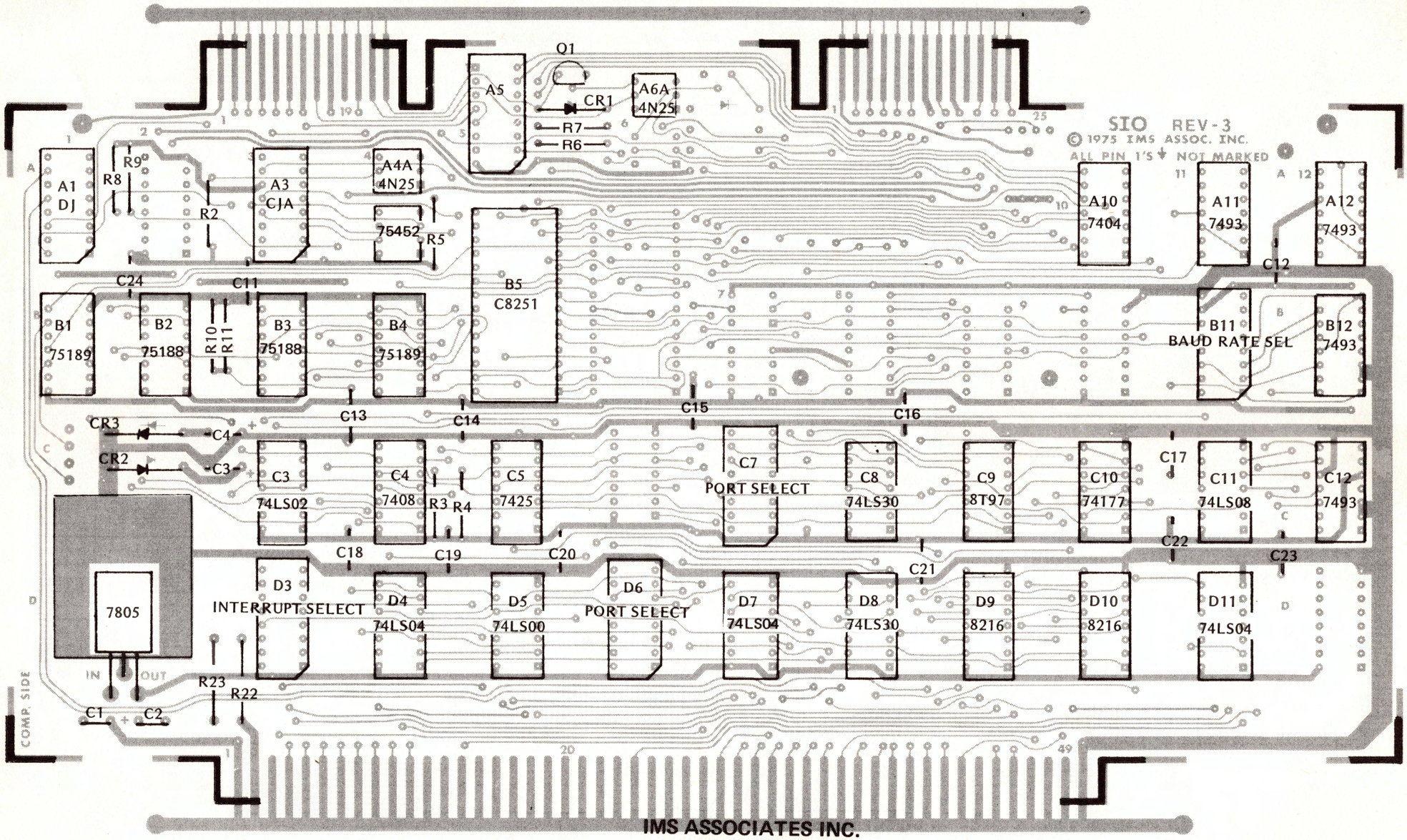
SIO 2-2, Rev. 3 91-1200018(K), 91-0200018(A)
Parts List

This package contains:

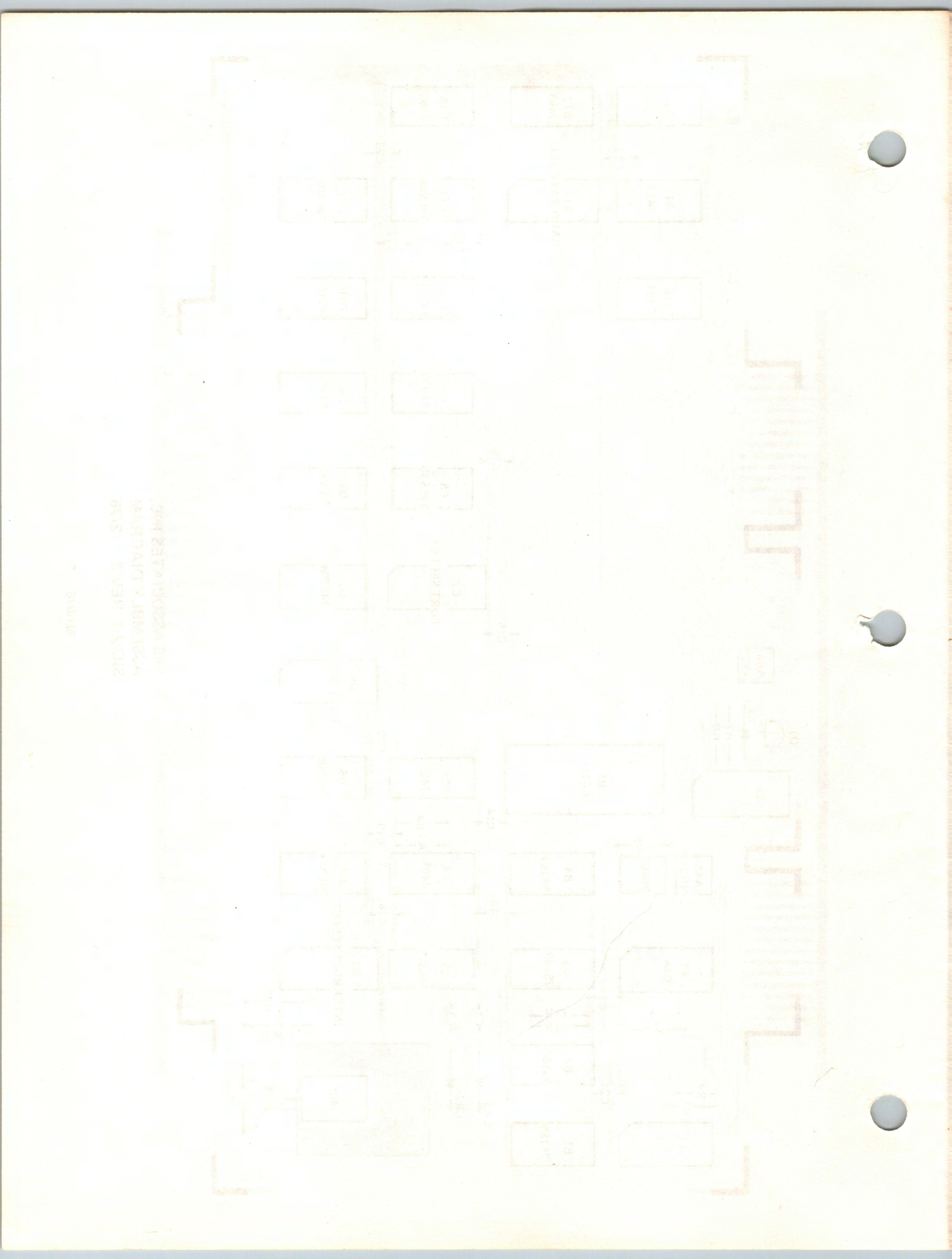
QUANTITY	IMSAI PART #	DESCRIPTION
10'	15-0000001	Solder
1	16-0100002	5 Prong Heat Sink, Thermalloy 6106-B-14
1	20-3302001	6-32x5/16" Phillips Pan Head Machine Screw
1	21-3120001	6-32 Hex Nut
1	21-3350001	#6 Internal Star Lockwasher
8	23-0400001	16 Pin IC Header, SAE c#6170-16
8	23-0800001	16 Pin Solder Tail IC Socket, TIN C831602
2	23-0800003	28 Pin Solder Tain IC Socket, TIN TIC832802
2	30-2560462	56 Ohm, ½ Watt, 5% Resistor/green, blue, black
6	30-3220362	220 Ohm, ¼ Watt 5% Resistor/red, red, brown
8	30-3470362	470 Ohm, ¼ Watt, 5% Resistor/yellow, violet, brown
7	30-4100362	1K Ohm, ¼ Watt, 5% Resistor/brown, black, red
2	30-4470362	4.7K Ohm, ¼ Watt, 5% Resistor/yellow, violet, red
16 14	32-2010010	.1uF, 30V Disk Capacitor
4	32-2233070	33-25 Tantalum Capacitor
2	35-1000006	1N914 Diode
2	35-1000009	1N4742 Zener Diode
2	35-2000002	2N3904 Transistor
4	36-0042501	IC, 4N25 Opto Isolator
1	36-0089701	IC, 8T97 (sub: 8097, 74367)
1	36-0740002	IC, 74LS00
1	36-0740202	IC, 74LS02
1	36-0740401	IC, 7404
3	36-0740402	IC, 74LS04
2	36-0740801	IC, 7408
1	36-0740802	IC, 74LS08
1	36-0742501	IC, 7425
2	36-0743002	IC, 74LS30
4	36-0749301	IC, 7493
1	36-0780501	IC, 7805 Regulator
2	36-0821601	IC, 8216
2	36-0825101	IC, 8251
1	36-7417701	IC, 74177
3	36-7518801	IC, 75188 1488
3	36-7518901	IC, 75189
2	36-7545201	IC, 75452
1	81-0000025	Chapter, SIO 2, Rev. 3
1	92-0000018	PC Board, SIO 2 Rev. 3

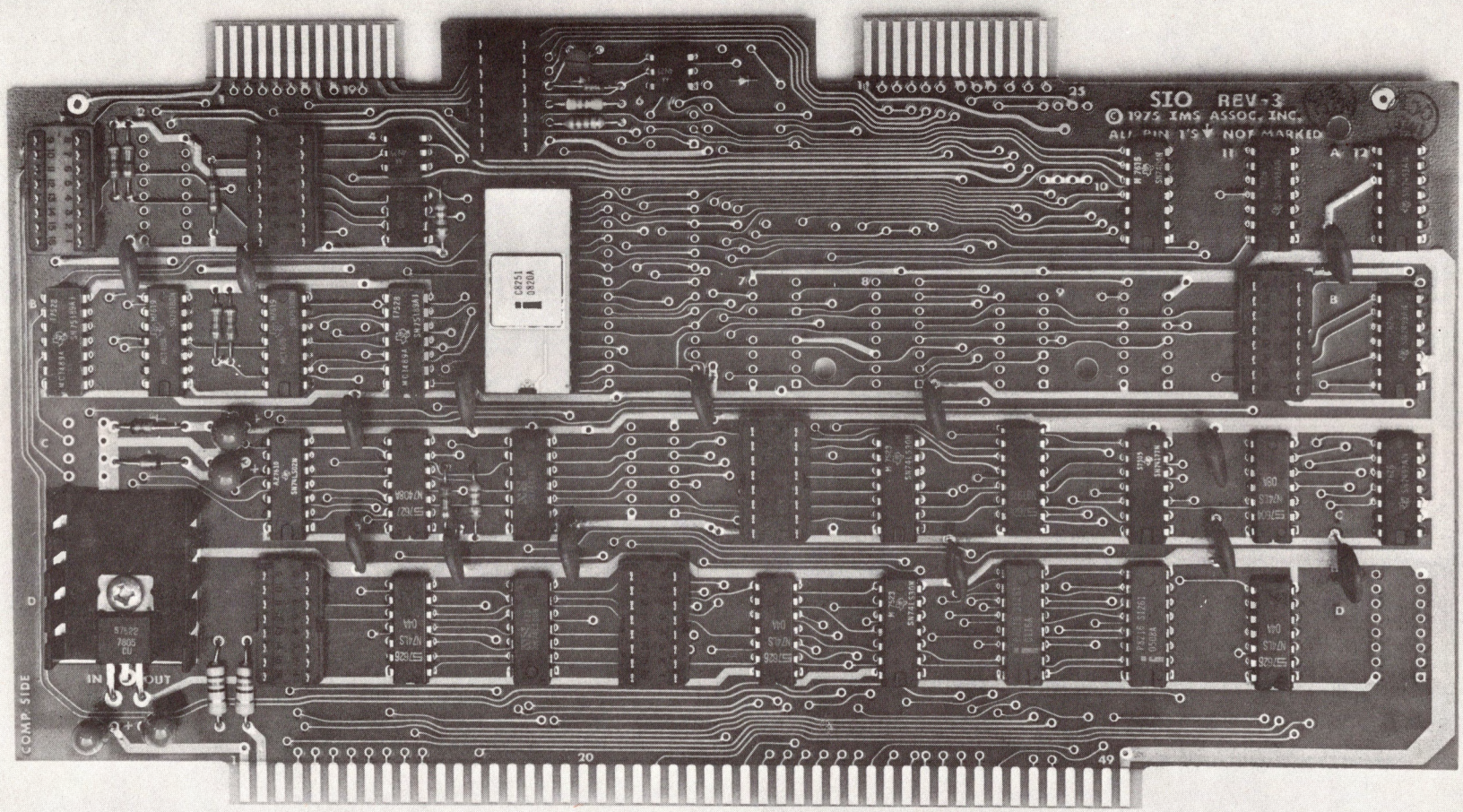
This is a revised parts list and may differ from that in your manual. Please use this list for parts check-off.

16/77

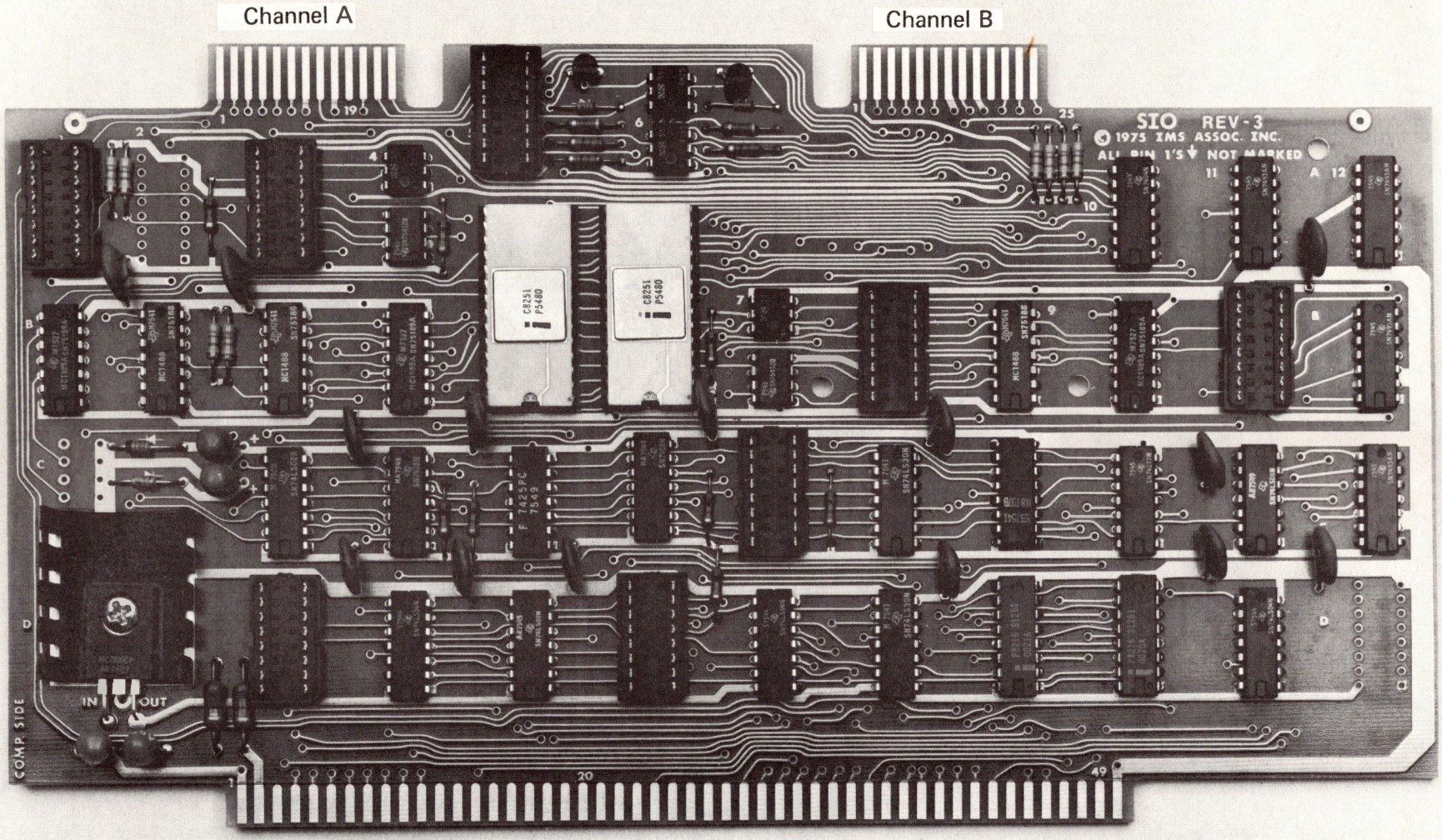


IMS ASSOCIATES INC.
ASSEMBLY DIAGRAM
SIO 2-1 REV 3 2/76





SIO 2–1 REV 3


Channel A
Channel B
SIO REV-3
© 1975 IMS ASSOC. INC.
ALL PIN 1'S NOT MARKED
C8251
P5480
C8251
P5480
F 7425PC
7549
IN
OUT
COMP SIDE

SIO 2-2 REV. 3

BOARD: SIO 2

ITEM	IMSAI PART #	QUANTITY	DESCRIPTION/IDENTIFYING MARKS
Solder	15-0000001	5'	
Heat Sink	16-0100002	1	Thermalloy/6106B-14
Screw	20-3402001	1	6-32x3/8" Phillips Pan Head Machine
Nut	21-3120001	1	6-32 Hex Nut
Lockwasher	21-3350001	1	#6 Internal Star Lockwasher
Header	23-0400001	7	16 Pin IC Header
Socket	23-0800001	7	16 Pin Solder Tail Socket
Socket	23-0800003	1	28 Pin Solder Tail Socket
Resistor	30-2560462	2	56 Ohm, ½ Watt/green. blue, black
Resistor	30-3220362	3	220 Ohm, ¼ Watt/red, red, brown
Resistor	30-3470362	4	470 Ohm, ¼ Watt/yellow, violet, brown
Resistor	30-4100362	4	1K Ohm, ¼ Watt/brown, black, red
Resistor	30-4470362	1	4.7K Ohm, ¼ Watt/yellow, violet, red
Capacitor	32-2010010	14	.1uF Disk Ceramic
Capacitor	32-2233070	4	33-25 Tantalum Capacitor (or 22-25)
Diode	35-1000006	1	1N914 Zener Diode
Diode	35-1000009	2	1N4742 Zener Diode
Transistor	35-2000002	1	2N3904 Transistor
Isolator	36-0042501	1	Opto Isolator/4N25
8T97	36-0089701	1	Hex Tri-State Buffer/N8T97B
74LS00	36-074002	1	Quad 2 Input NAND(Low Power Schottky)/ SN74LS00N
74LS02	36-0740202	1	Quad 2 Input NOR (LPS)/SN74LS02N

SIO 2-1 Rev. 3
Parts List

ITEM	IMSAI PART #	QUANTITY	DESCRIPTION/IDENTIFYING MARKS
7404	36-0740401	1	Hex Inverter/SN7404N
74LS04	36-0740402	3	Hex Inverter (LPS)/SN74LS04A
7408	36-0740801	1	Quad 2 Input AND/SN7408N
74LS08	36-0740802	1	Quad 2 Input AND (LPS)/SN74LS08N
7425	36-0742501	1	Dual 4 Input NOR with Strobe/SN7425N
74LS30	36-0743002	2	8 Input NAND (LPS)/SN74LS30N
7493	36-0749301	4	4 Bit Binary Counter/SN7493N
7805	36-0780501	1	5V Positive Voltage Regulator/MC7805CP
8216	36-0821601	2	Bi-Directional Bus Driver/D8216/S1261
8251	36-0825101	1	Programmable Comminication Interface/ C8251
74177	36-7417701	1	4 Bit Binary Counter, 35MHz/SN74177N
75188	36-7518801	2	RD 232 Driver/SN74188
74189	36-7418901	2	RS 232 Receiver/SN75189A
75452	36-7545201	1	Dual Peripheral Driver/SN75452BD
PC Board	92-0000018	1	SIO Rev. 3

BOARD: SIO 2

ITEM	IMSAI PART #	QUANTITY	DESCRIPTION/IDENTIFYING MARKS
Solder	15-0000001	5'	
Heat Sink	16-0100002	1	Thermalloy /6106B-14
Screw	20-3402001	1	6-32x3/8" Phillips Pan Head Machine
Nut	21-3120001	1	6-32 Hex Nut
Lockwasher	21-3350001	1	#6 Internal Star Lockwasher
Header	23-0400001	8	16 Pin IC Header
Socket	23-0800001	8	16 Pin Solder Tail Socket
Socket	23-0800003	2	28 Pin Solder Tail Socket
Resistor	30-2560462	2	56 Ohm, ½ Watt/green, blue. black
Resistor	30-3220362	6	220 Ohm, ¼ Watt/red, red, brown
Resistor	30-3470362	8	470 Ohm, ¼ Watt/yellow, violet; red
Resistor	30-4100362	7	1K Ohm, ¼ Watt/brown, black, red
Resistor	30-4470362	2	4.7K Ohm, ¼ Watt/yellow, violet, red
Capacitor	32-2010010	14	.1uF Disk Ceramic
Capacitor	32-2233070	4	33-25 Tantalum Capacitor (or 22-25)
Diode	35-1000006	2	1N914 Zener Diode
Diode	35-1000009	2	1N4742 Zener Diode
Transistor	35-2000002	2	2N3904 Transistor
Isolator	36-0042501	4	Opto Isolator/4N25
8T97	36-0089701	1	Hex Tri-State Buffer/N8T97B
74LS00	36-0740002	1	Quad 2 Input NAND (Low Power Schottky)/ SN74LS00N
74LS02	36-0740202	1	Quad 2 Input NOR (LPS)/SN74LS02N

SIO 2-2 Rev. 3
Parts List

ITEM	IMSAI PART #	QUANTITY	DESCRIPTION/IDENTIFYING MARKS
7404	36-0740401	1	Hex Inverter/SN7404N
74LS04	36-0740402	3	Hex Inverter (LPS)/SN74LS04A
7408	36-0740801	2	Quad 2 Input AND/SN7408N
74LS08	36-0740802	1	Quad 2 Input AND (LPS)/SN74LS08N
7425	36-0742501	1	Dual 4 Input NOR with Strobe/SN7425N
74LS30	36-0743002	2	8 Input NAND (LPS)/SN74LS30N
7493	36-0749301	4	4 Bit Binary Counter/SN7493N
7805	36-0780501	1	5V Positive Voltage Regulator/MC7805CP
8216	36-0821601	2	Bi-Directional Bus Driver/D8216/S1261
8251	36-0825101	2	Programmable Communication Interface/ C8251
74177	36-7417701	1	4 Bit Binary Counter, 35 MHz/SN74177N
75188	36-7518801	3	RD 232 Driver/SN75188
75189	36-7518901	3	RS 232 Receiver/SN75189A
75452	36-7545201	2	Dual Peripheral Driver/SN75452BD
PC Board	92-0000018	1	SIO Rev. 3

ITEM	IMSAI PART #	QUANTITY	DESCRIPTION/IDENTIFYING MARKS
7408	36-0740801	1	Quad 2 Input AND/SN7408N
75188	36-7518801	1	RD 232 Driver/SN75188
75189	36-7518901	1	RS 232 Receiver/SN75189A
75452	36-7545201	1	Dual Peripheral Driver/SN75452BD
8251	36-0825101	1	Programmable Communication Interface/ C8251
Diode	35-1000006	1	1N914 Zener Diode
Isolator	36-0042501	2	Opto Isolator/4N25
Header	23-0400001	1	16 Pin IC Header
Transistor	35-2000002	1	2N3904 Transistor
Resistor	30-3220362	3	220 Ohm, ¼ Watt/red, red, brown
Resistor	30-3470362	4	470 Ohm, ¼ Watt/yellow, violet, brown
Resistor	30-4100362	3	1K Ohm, ¼ Watt/brown, black, red
Resistor	30-4470362	1	4.7K Ohm, ¼ Watt/yellow, violet, red
Socket	23-0800001	1	16 Pin Solder Tail Socket
Socket	23-0800003	1	28 Pin Solder Tail Socket
Solder	15-0000001	5'	

SIO Assembly Instructions

1) Unpack your board and check all parts against the parts list enclosed in the package.

2) If gold contacts on the edge connector appear to be corroded, use pencil eraser to remove any oxidation. NOTE: Do not use Scotchbright or any abrasive material as it will remove the gold plating.

RESISTOR INSTALLATION

3) Insert and solder the two 56 ohm ½ watt resistors (green/blue/black) at locations R22 and R23 as shown on the Assembly Diagram.

4) Insert and solder the six 220 ohm ¼ watt resistors (red/red/brown) at locations R4 through R6, R12, R15, and R16 as shown on the Assembly Diagram for SIO 2-2; or the three 220 ohm ¼ watt resistors (red/red/brown) R4 through R6 for SIO 2-1.

5) Insert and solder the eight 470 ohm ¼ watt resistors (yellow/violet/brown) at locations R8 through R11 and R18 through R21 for SIO 2-2; the four 470 ohm ¼ watt resistors (yellow/violet/brown) at locations R8 through R11 for SIO 2-1 as shown on the Assembly Diagrams.

6) Insert and solder the 1K ohm ¼ watt resistors (brown/black/red) at locations R1 through R3, R13, and R14 for SIO 2-2; the 4 1K ohm ¼ watt resistors R2, R3, R13 and R14 for SIO 2-1 as shown on the Assembly Diagrams. The remaining two 1K ohm resistors for the SIO 2-2 board are used as jumper selects on A3 and B8. See User Guide section.

7) Insert and solder the two 4.7K ohm ¼ watt resistors (yellow/violet/red) at locations R7 and R17 shown on the SIO 2-2 Assembly Diagram; one 4.7K ohm ¼ watt resistor at location R7 as shown on the SIO 2-1 Assembly Diagram.

IC INSTALLATION

All Pin 1's are toward the lower right hand edge of the PC board and the 100 pin connector.

8) Insert and solder the one 74LS00 at location D5.

9) Insert and solder the one 74LS00 at locaiton D5.

10) Insert and solder the one 74LS02 at location C3.

11) Insert and solder the one 7404 at location A10.

12) Insert and solder the three 74LS04's at locations D4, D7, and D11.

13) Insert and solder the two 7408's at locations C4 and C6 on SIO 2-2; or the one 7408 at location C4 on SIO 2-1.

14) Insert and solder the one 74LS08 at location C11.

15) Insert and solder the one 7425 at location C5.

16) Insert and solder the two 74LS20's at locations C3 and D8.

17) Insert and solder the four 7493's at locations A11, A12, B12, and C12.

18) Insert and solder the one 74177 at location C9.

19) Insert and solder the three 75188's at locations B2, B3, and B9 on SIO 2-2; or the two 75188's at locations B2 and B3 on SIO 2-1.

20) Insert and solder the three 75189's at locations B1, B4, and B10 on SIO 2-2; or the two 75189's at locations B1 and B4.

21) Insert and solder the two 75452's at locations A4B and B7B on SIO 2-2; or the one 75452's at location A4B on SIO 2-1.

22) Insert and solder the two 8216's at locations D9 and D10.

23) Insert and solder the two 28 pin solder tail sockets at locations B5 and B6 on SIO 2-2; or the one 28 pin solder tail socket at location B5.

24) Insert and solder the four 4N25's at locations A4A, A6A, A6B, and B7A for SIO 2-2; or the two 4N25's at locations A4A and A6A for SIO 2-1.

25) Insert the one 8T97 at location C9.

DO NOT copy

DISCRETE COMPONENT INSTALLATION

26) Insert and solder the fourteen .1 uf disk capacitors at locations C11 through C24 as shown on the Assembly Diagram.

27) Insert and solder the four 33 uf tantalum capacitors at locations C1 through C4 as shown on the Assembly Diagram. NOTE: Observe polarity (+ to +) as shown on the board.

28) Insert and solder the two 1N914 zener diodes at locations CR1 and CR4 as shown on the Assembly Diagram.

29) Insert and solder the two 1N742 zener diodes at locations CR2 and CR3 as shown on the Assembly Diagram.

30) Insert and solder the two 2N3904 transistors at locations Q1 and Q2 as shown on the Assembly Diagram for SIO 2-2; or the one 2N3904 transistor at location Q1 as shown on the Assembly Diagram for SIO 2-1.

31) Insert and solder the eight 16 pin sockets at locations A1, A3, A5, B8, B11, D3, and D6 for SIO 2-2; or the seven 16 pin sockets at locations A1, A3, A5, B11, D3, and D6 for SIO 2-1 as shown on the Assembly Diagram.

REGULATOR AND HEAT SINK INSTALLATION

32) Before installing the heat sink and regulator, bend the 7805 regulator leads at 90 degree angles to facilitate mounting of the heat sink.

33) Insert the #6 screw through the 7805 regulator and heat sink on the component side of the board and attach through the lockwasher and nut on the circuit side of the board. Tighten the screw carefully to insure proper alignment of the heat sink to prevent shorting to adjacent traces Solder in the 7805 leads.

34) Finally, the 8251 chips should be inserted in their sockets with Pin 1 down toward the 100 pin edge connector at the bottom of the board. Addressing and baud rate jumpers should be installed and other option jumpers installed as required. The board is ready for use.

USER GUIDE

The IMSAI SIO Board provides 2 independent channels of serial data input and output. Utilizing the Intel 8251 USART devices, the SIO Board provides 2 channels of RS232, TTL, and current loop data lines with complete control signals.

The SIO Board also includes all logic necessary to control the 8251 devices from the IMSAI 8080 Back Plane.

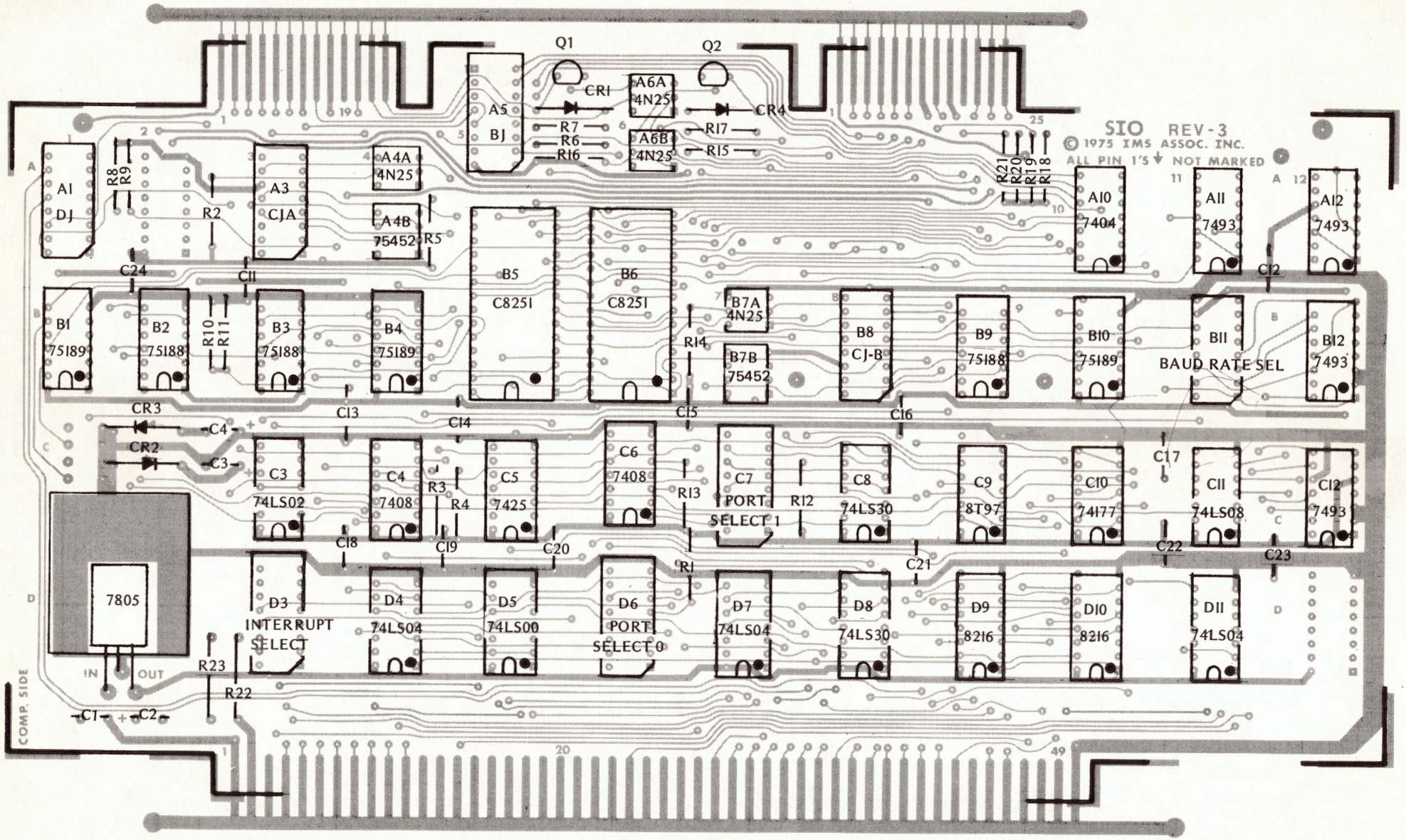
For reference information on the programming and operation of the 8251 chip, the user should refer to the Intel 8080 Microcomputer Systems User's Manual.

The User's Guide is intended to cover the information beyond that contained in the Intel Data Book necessary to make full use of the SIO board.

Both the memory-mapped and jumper-wired I/O configurations use the lower 4 bits of the address bytes (A1 through A3) to select and control the board's functions. Bit 4 through 7 of the board address (A4 - A7) are jumper-selected as described on another page. If the board is jumper-selected to run as an input and output port type board, then A0 - A7 form a complete address. If the board is jumper-selected to respond to memory-mapped I/O, then A0 - A7 form the lower byte of address and the upper byte of address is hex FF or octal 376.

Address bits 1 and 2 select serial I/O channel A or channel B respectively. That is, when address bit 1 (A1) is high, serial I/O channel A is enabled. When address bit 2 (A2) is on, serial I/O channel B is enabled.

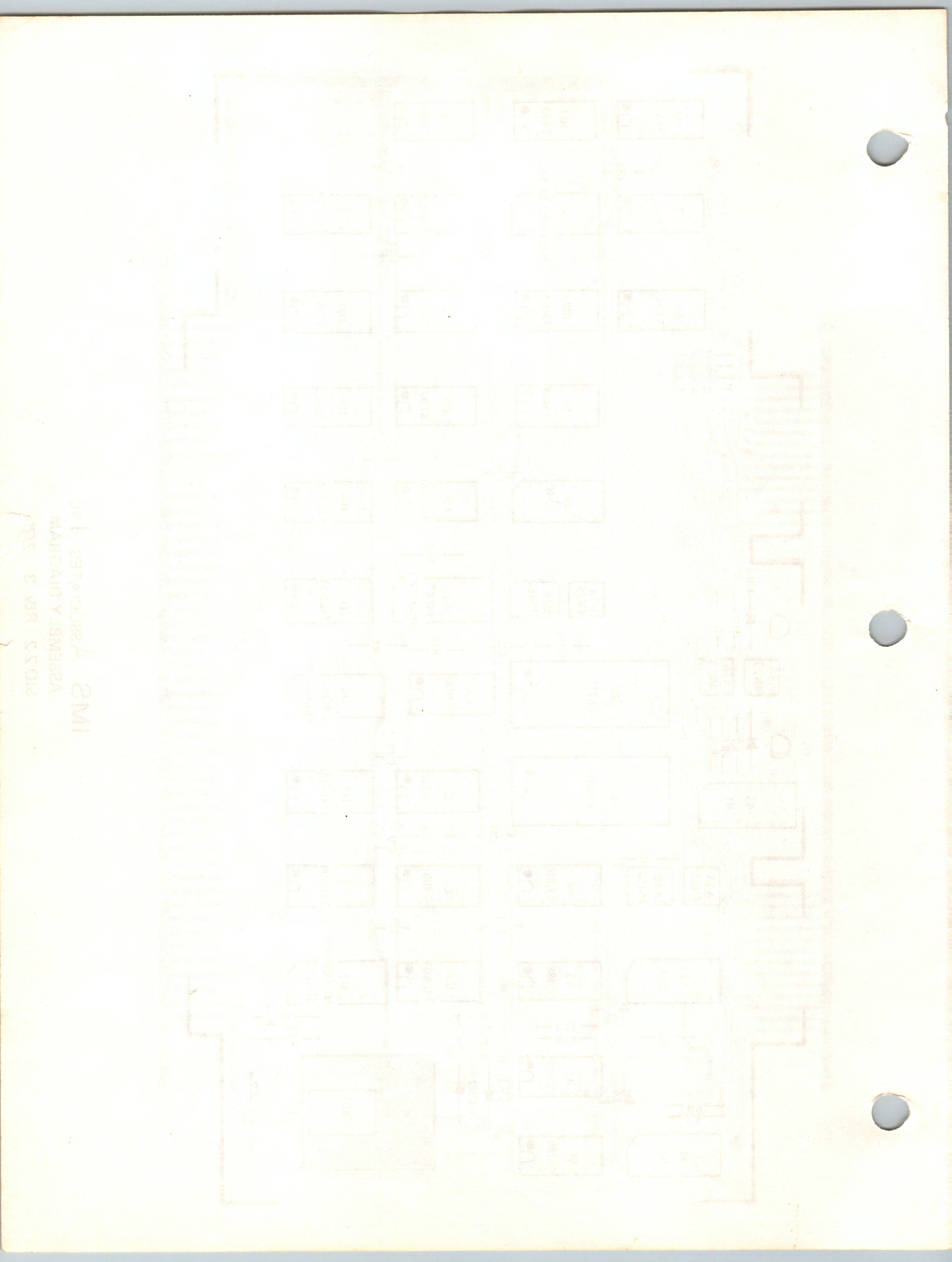
Address bit 0 determines whether the I/O channel selected will respond to the current byte as a control byte or a data byte. If address bit 0 is a 1, the control functions are selected, and if address bit 0 is a 0, the byte is assumed to be data.. Thus, to write a control byte into serial I/O channel A, the lower 4 bits of address would normally contain hex 3 or octal 03, while the normal address



IMS Associates Inc.

ASSEMBLY DIAGRAM

SIO 2-2 REV 3 2/76



SIO BOARD ADDRESSING

Address Bit	Function	
0	C/$\overline{D}$ on 8251's	1 = CONTROL 0 = DATA
1	SELECT CHANNEL A	1 = SELECT
2	SELECT CHANNEL B	1 = SELECT
3	SELECT CONTROL I/O	1 = SELECT
4		
5	CARD ADDRESS	
6	Jumperable to any one of 16 addresses	
7		

I/O DATA PORT A = 02H
I/O DATA PORT B = 04H
I/O Control Port A = 03H
I/O Control Port B = 05H

This byte is I/O port address to run SIO card from INP & OUT instructions.

If SIO card is to be run from memory reference instructions (memory mapped I/O), the above byte is the low order address byte; the high order address byte is FE_{hex} (376_{octal}) (1111 1110 binary)

SIO CONTROL I/O BIT DEFINITIONS

Bit	Input Byte	Output Byte
0	always 1	Interrupt Enable chan. A
1	always 1	Carrier Detect chan. A
2	Carrier Detect chan. A	non - functional
3	Clear To Send chan. A	non - functional
4	always 1	Interrupt Enable chan. B
5	always 1	Carrier Detect chan B
6	Carrier Detect chan. B	non - functional
7	Clear To Send chan. B	non - functional

Carrier detects need option jumper to select originate/receive

Interrupts occur on TxRDY, TxEMTY, RxRDY, and SYNDET

TxRDY and RxRDY interrupts are removed if the respective functions (transmit and receive) are disabled by software command byte. TxEMTY interrupt is removed only by filling transmit buffer with a byte. This may be done while the transmit function is disabled .if desie

for channel B control bytes would be hex 5 or octal 05. Address bit 3 (A3) selects the board control I/O port. When address bit 3 (A3) is high, the control port will be enabled. Thus, when use is being made of the control port, the lower 4 bits of address would normally be hex 8 or octal 10.

The control I/O byte selected by address bit 3 is divided into the upper 4 bits and the lower 4 bits. The lower 4 bits, 0 through 3, serve the channel A serial I/O circuit. The upper four bits, 4 through 7, serve the second I/O channel B functions. Bits 0 and 4, for channel A and B respectively, control the interrupt enable separately for each channel. When this bit is a 1, the interrupts are enabled and the processor will receive and interrupt whenever any one of the following 4 lines are active: the transmitter ready line, the transmitter empty line, the receiver ready line, and the sync detect line.

If bits 0 or 4 (as appropriate to channel A or B) are made 0, then no interrupts will be generated from the affected channel. Bits 1 and 5 serve channel A and B, respectively, to output the carrier detect signal. This is operative only when the jumper in jumper socket BJ has selected the board to act as the originator of the carrier detect line.

Bits 2, 3, and 6, and 7 are not functional in the output mode for the SIO control byte. When an input is read from the SIO control byte, bits 0, 1, 4 and 5 are not functional. These 4 bits will always be read as a 1.

Bits 2 and 6 read the condition of the carrier detect receiver for channels A and B, respectively. The signal is operative only when jumper socket BJ is jumpered to read the condition of the carrier detect line.

Bits 3 and 7 serve channel A and B, respectively, to read the condition of the clear-to-send (CTS) control signal. This is provided because it is not possible to read the condition of CTS through programmed input from the 8251.

SIO BOARD I/O PIN DEFINITIONS

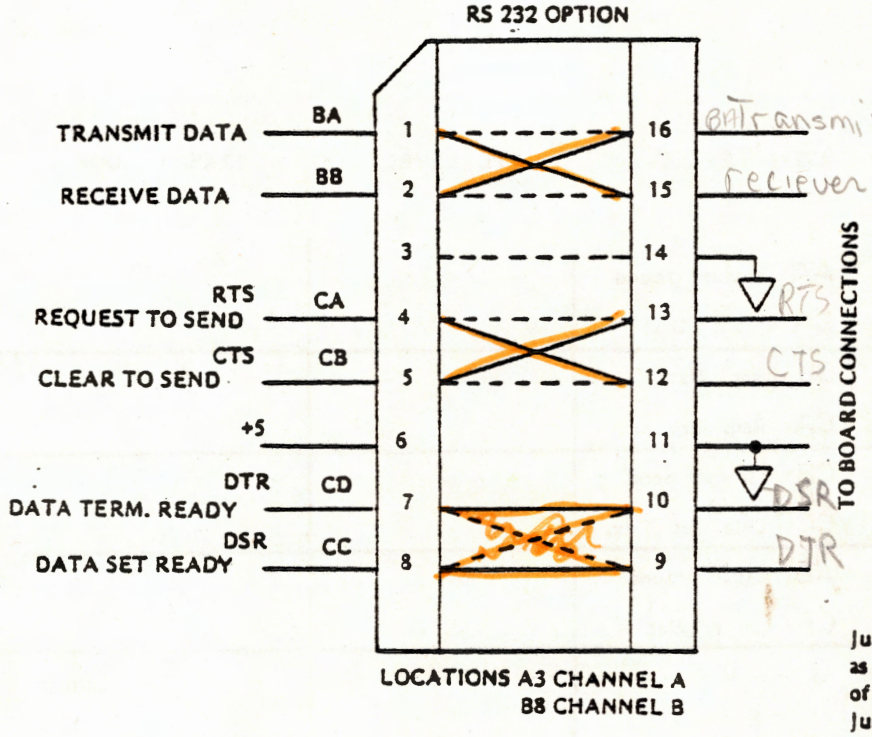
from TTY (Data Out) Red + J4 1
Brown − J4 12

to TTY (Data In) Orange + J4 10
Yellow − J4 3

TTY Tape Reader Blue +
Green −

EIA 25 pin connector	26 pin edge connector	RS232 LEVELS	TTL LEVELS	CURRENT LOOP
1	1	AA chassis ground		
2	3	BA Trans. Data		
3	5	BB Rec. Data		
4	7	CA Req. to Send		
5	9	CB Clr. to Send		
6	11	CC Data Set Rdy.		
7	13	AB signal ground		
8	15	CF Carrier Det.		
9	17	+ V		+ V + Current Source orange
10	19	Jumper		
11	21			In Loop +
12	23			Out Loop + Yellow
13	25	Jumper		Out Loop −
14	2		Data Term. Rdy.	
15	4	DB Trans. Clk.		
16	6		Data Set Rdy.	+12
17	8	DD Rec. Clk.		
18	10		Data Out	
19	12		Data In	
20	14	CD Data Term. Rdy.		
21	16			Current sink 1
22	18			
23	20			Current sink 2 Brown Blue
24	22			
25	24			In Loop − Red Green

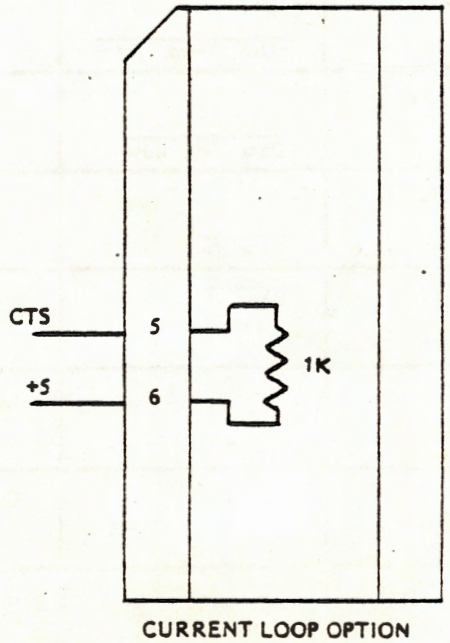
Ch A Ch B



Jumpers shown for connection as terminal or computer end of an RS232 line.
Jumper connection 3 to 14 is always to be made.

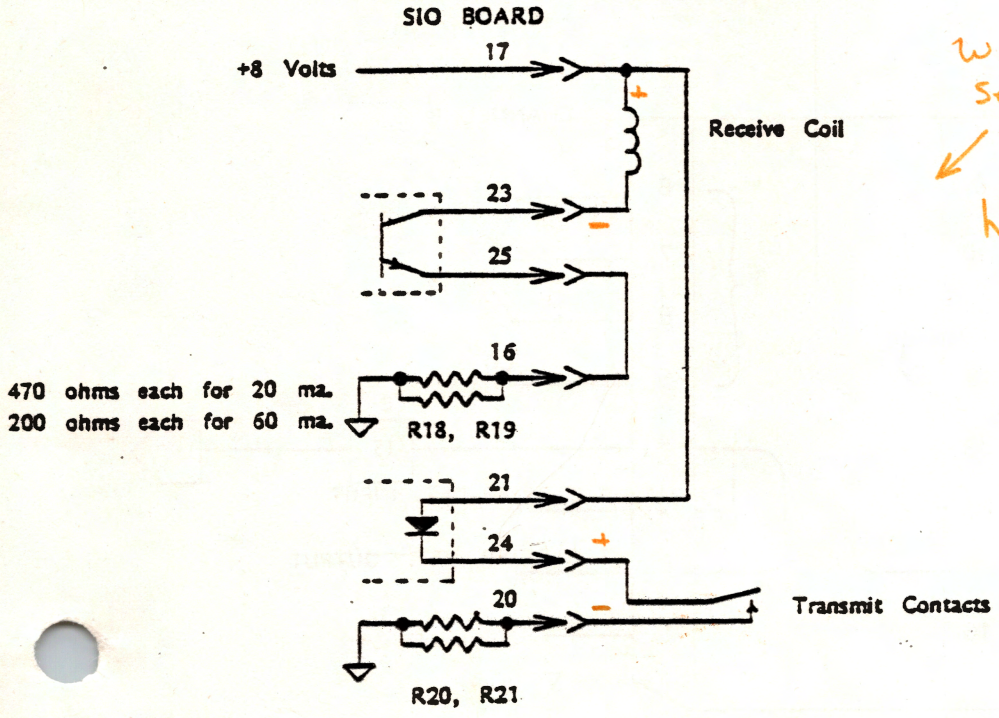
TERMINAL ——

COMPUTER ——



CURRENT LOOP CONNECTIONS

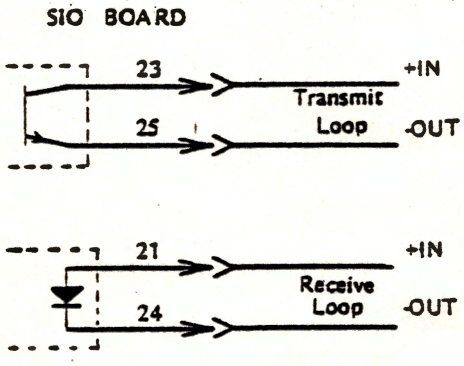
WITH CURRENT SOURCE



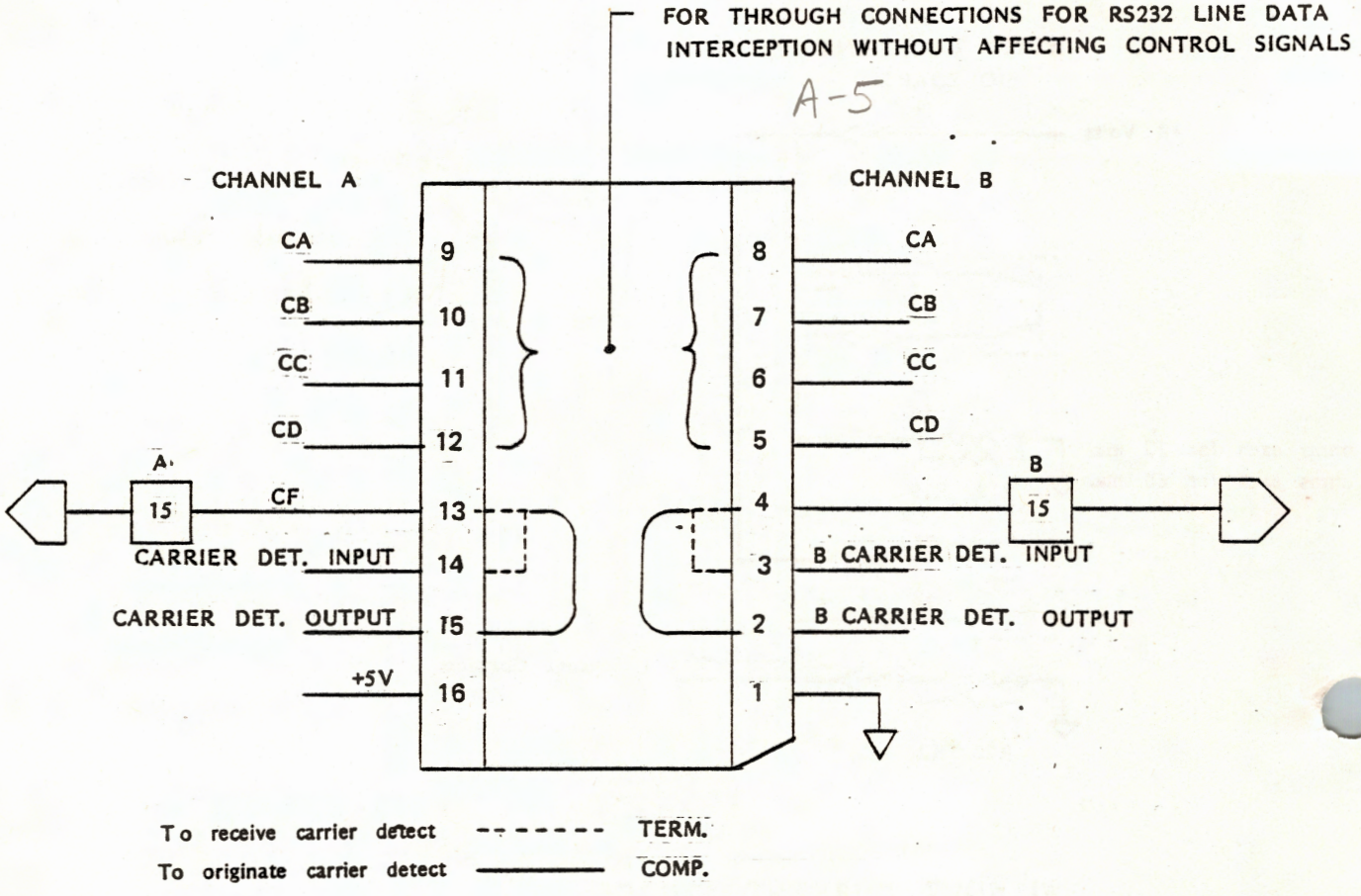
we use this set up

Channel two hard wired on board

WITHOUT CURRENT SOURCE



SIO RS232 INTERCHANNEL CONTROL JUMPERS and CARRIER DETECT



The TTL output levels are driven by a 75452 dual peripheral driver, with open collector outputs, and a 220 ohm pull-up to +5 volts. The TTL data inputs drive 1TTL input load and a 1K pull-up to +5 volts.

When the TTL inputs are not being used, they should be left open or held high so as not to affect data input from other sources.

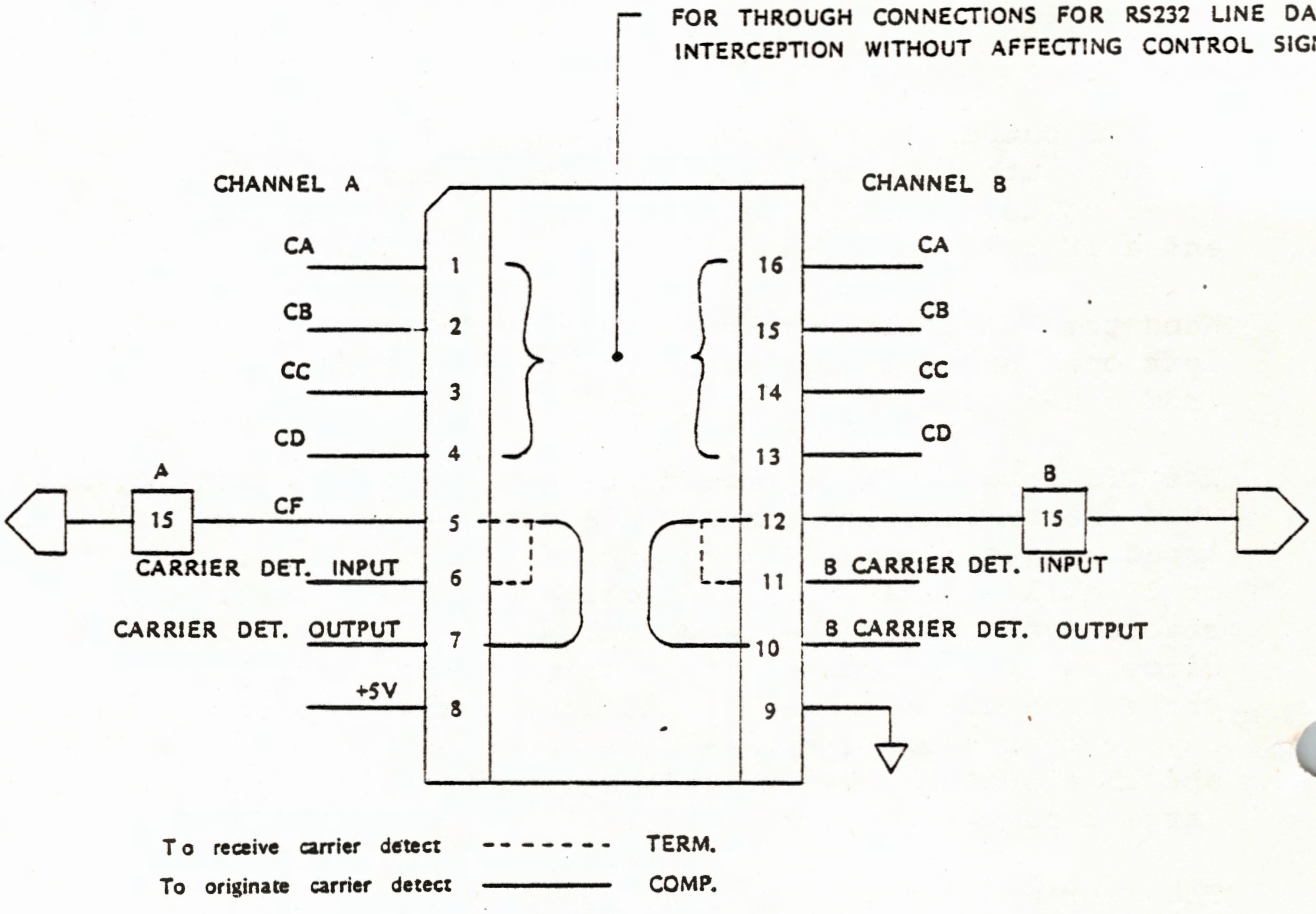
The TTL Data Input line must be left open and not held high when the current loop inputs are used. The current loop input drives opto-isolators and will respond to either 20 or 30 milliamperes. In applications where a significant reverse voltage may be experienced, such as when inductive circuits (i.e., relays) are coupled to the data line, a protective diode should be put across the line such that any reverse voltage spikes will cause the diode to conduct and thus protect the LED in the opto-isolator from too large a reverse voltage.

The current loop output is switched by an isolated transistor through an opto-isolator and is provided with a transient-shunting diode across the output transistor so that it may be used to drive relays without risk of damage to the output circuit. Typical wiring connections are diagramed on another page, both with and without the current source being used.

Setting the baud rate for serial I/O channels A and B is done on the jumper select socket RJ in position B11. The baud rates designated on the detailed sheet for rate select are correct when the 8251 is programmed for a 16X asynchronous clock rate and a 1X synchronous clock rate.

The details of selecting the desired baud rates are located on the schematic.

SIO RS232 INTERCHANNEL CONTROL JUMPERS and CARRIER DETECT



Jumper CJ-A or CJ-B

The jumper selection socket in A3 serves serial I/O channel A and the jumper selection socket in B8 serves serial I/O circuit B. Their functions are the same for their respective channels. The function of this jumper socket is to permit the serial I/O port RS232 levels to be wired so as to either serve as the terminal end of a 232 line or the computer end of a 232 line with no special cable wiring required off the Serial I/O board.

With pins 1, 2, 4, 5, 7 and 8 wired directly across the jumper socket as shown in the diagram for the terminal end, the function of the lines correspond one to one with the names of the RS232 control lines referred to in the 8251 specifications.

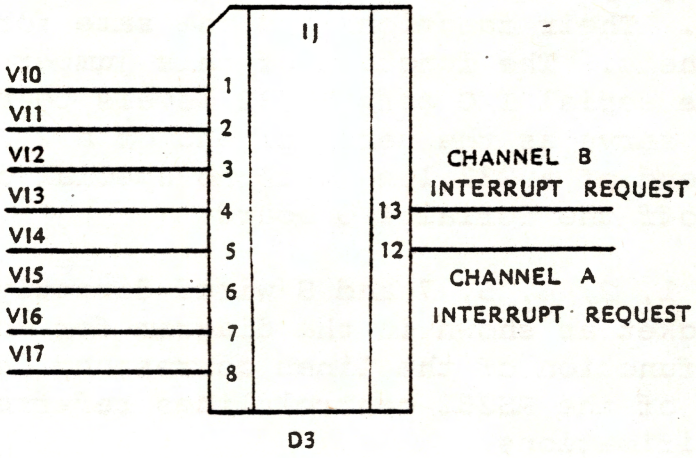
The inputs and outputs are arranged as appropriate for the SIO board to serve as the terminal end of an RS232 line. Should it be desired for the SIO board to serve as the computer end of a standard RS232 line, use jumpers connected as shown in the diagram. The 3 pairs of lines are reversed so that TRANSMIT DATA is now driving what is received data for the terminal and RECEIVE DATA is receiving what is transmit data from the terminal, and similarly, REQUEST TO SEND and CLEAR TO SEND are reversed and DATA SET READY and DATA TERMINAL READY are reversed.

Ground and +5 volts are available on the socket for providing permanent mark or space levels to any of the control lines if CLEAR TO SEND is not driven by an external source. It should be wired to pin 6 to provide a constant enable for the transmitter section of the USART.

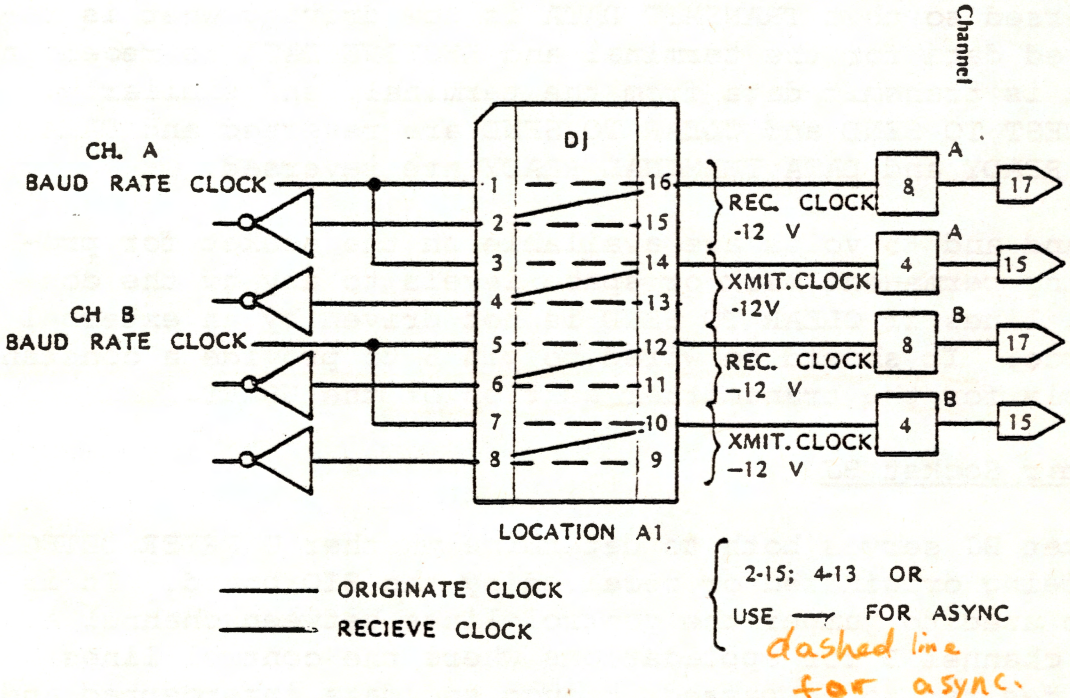
Jumper Socket BJ

Socket BJ serves both to determine whether CARRIER DETECT is being originated or received by the SIO board. It is also used to jumper the control lines between channel A and channel B for applications where the control lines are desired to be passed through and data intercepted and

SIO INTERRUPT SELECT SOCKET



SIO 232 CLOCKS JUMPER OPTIONS



Program 8251 for x16 for asynchronous operation,
x1 for synchronous.

handled. The four primary control lines for both channel A and channel B appear in this jumper socket, and can be jumper-wired straight across as desired.

It should be remembered that only one source should be driving an RS232 line at a time. If the control lines are jumpered straight across so that the modem and data terminal are driving the lines, then appropriate jumpers in jumper socket locations A3 or B8 should be removed so that the SIO board will not be attempting to drive these lines at the same time. If it is desired to detect the DATA TERMINAL READY line, then a jumper needs to be placed as shown between pins 5 and 6 for channel A, or between pins 11 and 12 for channel B.

If it is desired to originate the CARRIER DETECT line, a jumper should be placed instead between pins 5 and 7 for channel A, for 10 and 12 for channel B.

Ground and +5 volts are available in this jumper socket for providing a permanent mark or space level to any of these control lines.

The interrupt line from channel A and channel B both appear on the interrupt select socket in position D3. All 8 of the IMSAI 8080 system priority interrupt lines on the back plane, also appear on the interrupt select socket. A jumper may be placed between the appropriate channel's interrupt line and any one of the priority interrupt system lines to provide an interrupt of the desired priority.

Jumper Location DJ, Located in A1

The jumper select socket in A1 provides facilities for originating and receiving clock signals for receive or transmit for use in the synchronous mode of communication. One-half of the socket controls lines for Channel A and the other half is dedicated to Channel B. Pins 1, 2, 3, 4, and 13, 14, 15 and 16 serve the channel A jumper functions. The remainder of the pins have the identical function for Channel B.

When it is desired to originate the clock signal the pins for that channel should be jumpered straight across, as shown in the diagram, so that the clock signal from the SIO board is driven through converters to RS232 levels onto the DD and DB lines.

The inputs to the data clock receive circuits are tied to -12 volts to provide an inactive output to the OR-gate supplying the receive clock to the USART chip.

When it is desired instead to receive the clock from the RS232 cable, then these jumpers are removed and the RS232 lines DD and DB are jumpered to the input of the clock-receive circuits as shown in the diagram.

When this is done, the data rate select socket for the appropriate channel must be jumpered so that the clock line from this jumper select socket is held at ground or low in order to avoid interference between the onboard clock circuit and the incoming clock from the RS232 line.

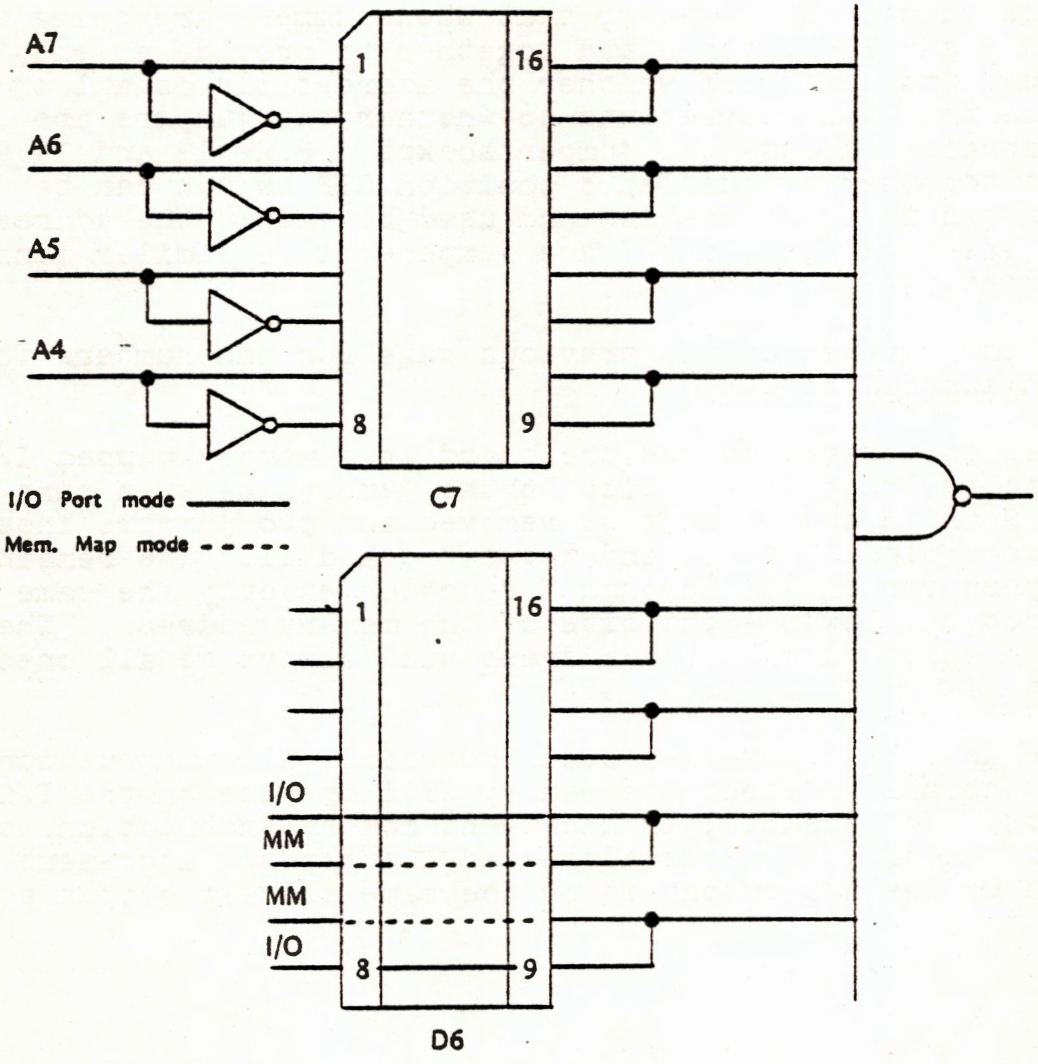
Data Rate Select Socket

The jumper socket in position B11 provides for selecting different baud rates for both Channel A and Channel B from the set of standard rates provided by the SIO board. The pin numbers and baud rates are indicated in the diagram.

The clock lines for Channel A and Channel B are completely independent and may be jumpered to the same rate or different rates.

When the chip is being used in the synchronous mode, the chip is running at a 1X clock rate rather than 16 X rate as in the asynchronous mode. Thus, the baud rates are 16 times as great for the same jumper location when used in the synchronous mode.

Board Address Selection Jumper Sockets



The board address is selected by jumpers or a DIP switch in locations C7 and D6. There are two cases for which this board may be jumpered: 1) to respond to input/output instructions and 2) to respond to memory access instructions. The case of input/output instructions will be treated first.

In selection location D6 pins 8 and 9 must be jumpered together and pins 5 and 12 must be jumpered together. The user must jumper socket C7 so when the desired I/O Port Address appears on the Address lines, the inputs to the NAND gate from bits A4 through A7 are high. If, for instance, address bit 6 is desired to be a 0 when the board responds, then pins 4 and 13 would be jumpered together. If address bit A6 was desired to be a 1, then

either pins 3 and 14 may be jumpered together or 3 and 13 may be jumpered together, since 13 and 14 are tied to the common address selection input.

It is suggested, however, that when jumpers are being used, pins 3 and 13 be connected together to provide an easy visual indication of whether the address bit is a 1 or a 0. since that will correspond to whether the jumpers are slanted or straight across the jumper socket. Pins 13 and 14 were tied together so that an 8 position DIP switch can be inserted in this location and used to select the address. Address bits 4, 5, and 7 are jumpered in a similar manner on position C7.

See the diagram on the previous page for pin numbers for each address bit.

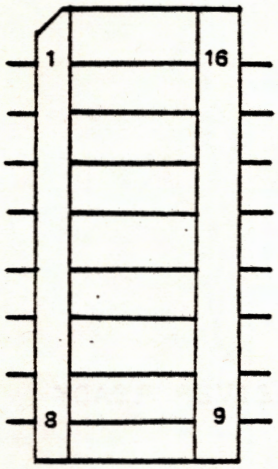
If it is desired to use the board in a memory-mapped I/O capacity, then in position D6 the jumpers between pins 8 and 9 and 5 and 12 must be removed and two jumpers inserted between pins 7 and 10 and between 6 and 11. The remaining jumpers for bits 4 through 7 function exactly the same and affect the lower eight bits of the memory address. The upper eight bits of the address will always be all ones, that is hex FE or octal 376.

When used as a memory-mapped I/O board, all instructions that normally affect the memory will operate on the I/O ports. For example, an increment memory instruction would read the data from the addressed input port, increment that data by one and output it on the same address output port.

Example Jumpers -

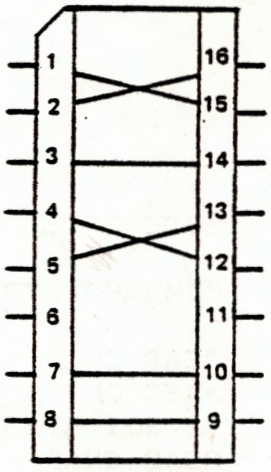
To use the SIO Board in its simplest form, non-interrupted input/output instruction controlled, create jumpers as shown.

A1

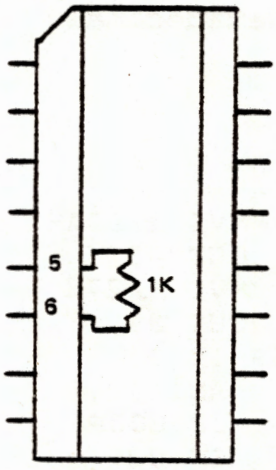


RS232 or Current Loop

A3 (B8)

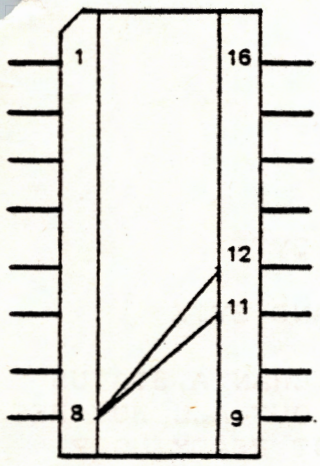


RS232

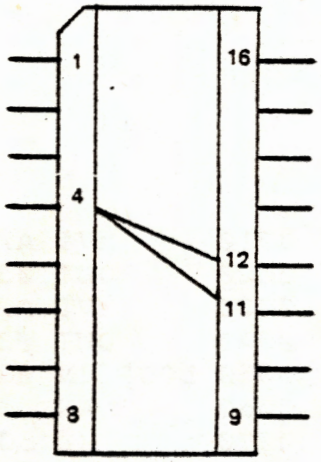


Current Loop

B11



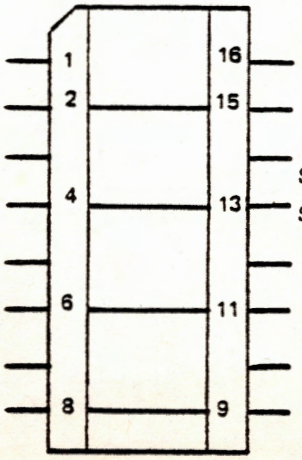
110 BAUD



1200 BAUD

D6

C7



Select appropriate address-
Shown is 0- as required by SCS

Sample sequence to set up SIO for teletype and echo from keyboard to printer:

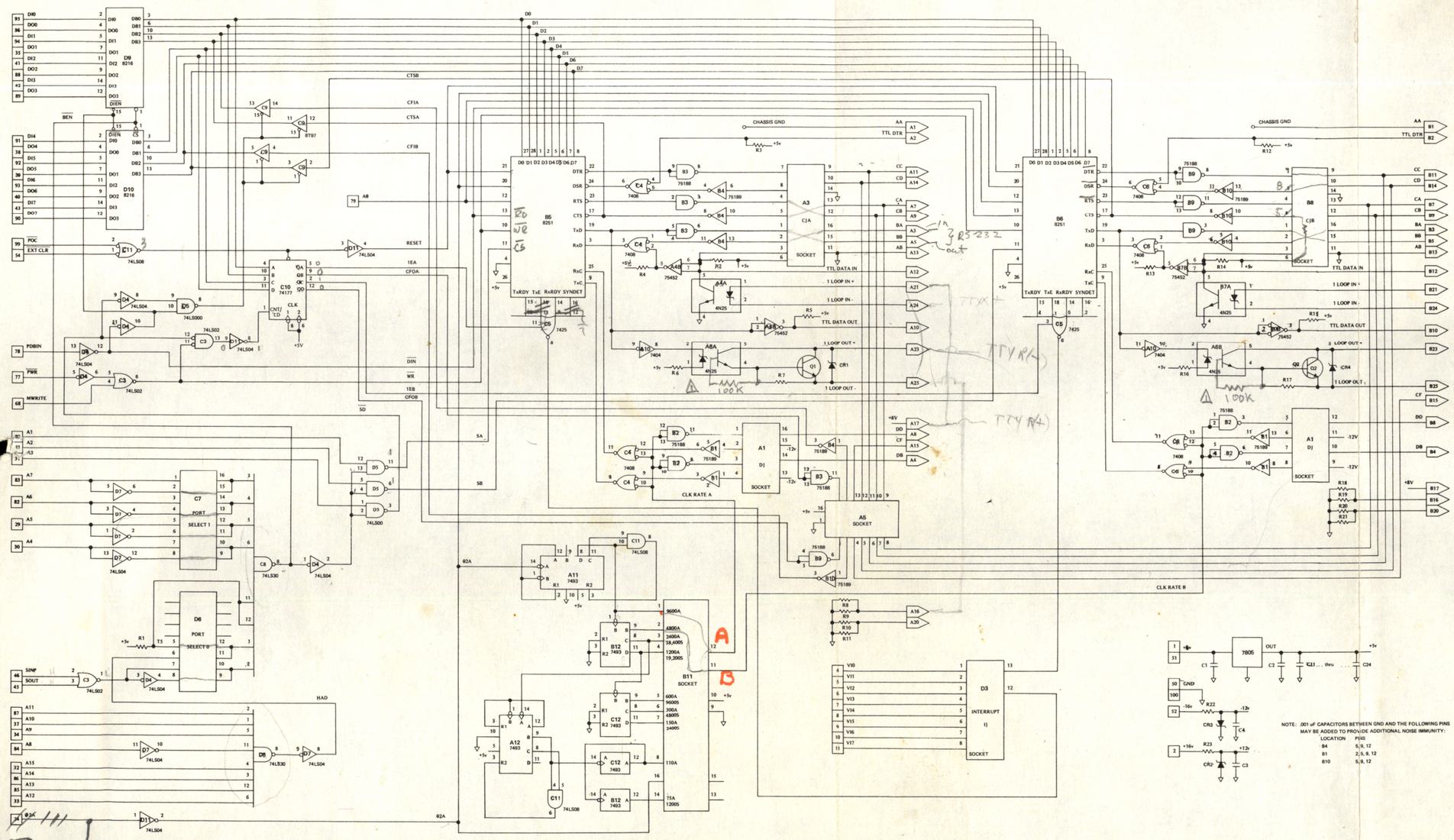
Format used is 2 stop bits, no parity, and 7 data bits. Reset IMSAI 8080 before running. Address and constants are in hexadecimal.

```
LIST
0010      MVI A,0CAH          MODE BYTE
0020      OUT 03
0030      MVI A,27H           COMMAND BYTE
0040      OUT 03
0050 LOOP IN 03               READ CHAN A STATUS
0060      ANI 02              MASK OUT ALL BUT RECEIVER READY
0070      JZ LOOP             IF NOT READY LOOP
0080      IN 02               READ CHAR
0090      OUT 02              WRITE CHAR
0100      JMP LOOP
```

```
ASSM 3700

3700 3E CA      0010      MVI A,0CAH          MODE BYTE
3702 D3 03      0020      OUT 03
3704 3E 27      0030      MVI A,27H           COMMAND BYTE
3706 D3 03      0040      OUT 03
3708 DB 03      0050 LOOP IN 03               READ CHAN A STATUS
370A E6 02      0060      ANI 02              MASK OUT ALL BUT RECEIVE
370C CA 08 3701 0070      JZ LOOP             IF NOT READY LOOP
370F DB 02      0080      IN 02               READ CHAR
3711 D3 02      0090      OUT 02              WRITE CHAR
3713 C3 08 3701 0100      JMP LOOP
```

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED
0	ORIGINAL REV. 3	5/76	
1	UPDATE & MOD.	12/76	



A1, A3, A5	OPTIONAL JUMPER SOCKET	R1, R2, R3, R13, R14 — 1K ¼W
A4B	75452	
A4A, A6A, A6B	4N25	R4, R5, R6, R12, R15, R16 — 220 ¼W
A10	7404	
A11, A12	7493	
B1, B4, B10	75189	R7, R17 — 4.7 ¼W
B2, B3, B9	74188	R8, R9, R10, R11, R18, R19, R20, R21 — 470 ¼W
B5, B6	8251	
B7A	4N25	
B7B	75452	
B8, B11	OPTIONAL JUMPER SOCKET	R22, R23 — 56 ½W
B12	7493	
C3	74LS02	
C4, C6	7408	
C5	7425	
C7	PORT SELECT 1	
C8	74LS30	
C9	8T97	
C10	74177	
C11	74LS08	
C12	7493	
D3, D6	OPTIONAL JUMPER SOCKET	
D4, D7, D11	74LS04	
D5	74LS00	
D8	74LS30	
D9, D10	8216	
C1 thru C4	33 uF	
C11 thru C24	.1 uF	
CR1, CR4	1N914	
CR2, CR3	1N4742	
Q1, Q2	2N3904	

EIA — J4

9 – A17 — 10 – Org – TTY R(+)
23 – A20 — 12 – Bn – TTY X (-)
12 – A23 — 3 – Yel – TTY R(-)
25 – A24 — 1 – Red – TTY X (+)
(+12) 16 – A6 — 5 – Bl (+)
(GND) 7 – A13 — 8 – Grn (-)

9600 Baud Mod. 6/30/78

SR 10-15-80 Runs with any up speed.

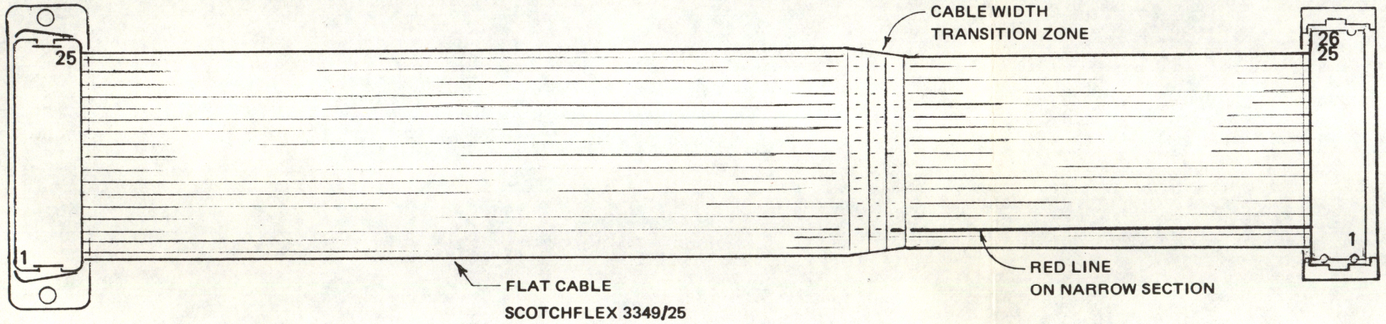
© 1977 IMSAI MFG. CORP. SAN LEANDRO, CA. ALL RIGHTS RESERVED WORLDWIDE MADE IN U.S.A.

TOLERANCES UNLESS OTHERWISE SPECIFIED FRACTIONS DEC. ANGLES ± ± ±	IMSAI MFG. CORP. SAN LEANDRO, CA.		
APPROVALS	DATE	SCHEMATIC DIAGRAM SIO 2–2 REV. 3 12/76	
DRAWN			
CHECKED		SCALE	SIZE / DRAWING NO 91–4010018
		DO NOT SCALE DRAWING	SHEET

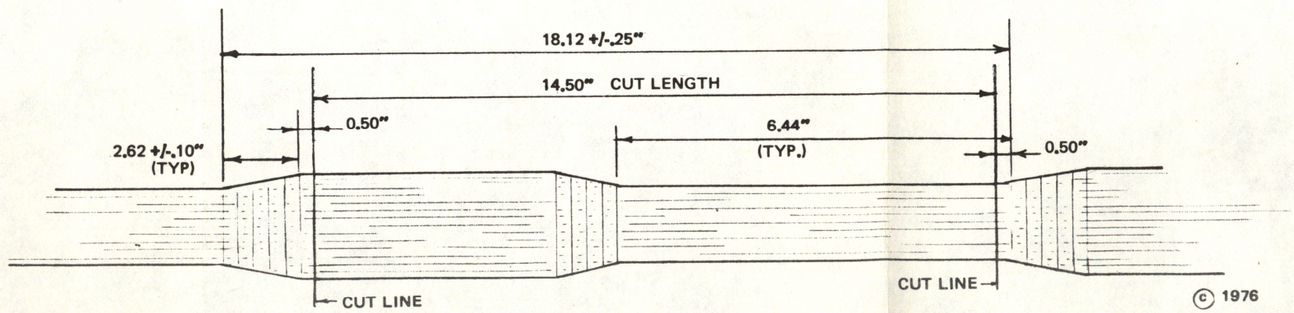
BISHOP GRAPHICS, INC. REORDER NO. 20621

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED
0	ORIGINAL	8/76	
1	REVISED CABLE DIMENSIONS	8/76	

25 PIN FEMALE CONNECTOR
SCOTCHFLEX 3483-1000
MOUNTING HARDWARE SET
SCOTCHFLEX 3341-1



26 PIN SOCKET CONNECTOR
SCOTCHFLEX 3462-0001



CABLE DETAIL

© 1976
IMS ASSOCIATES INC.
SAN LEANDRO, CA.

© 1977 IMSAI MFG. CORP.
SAN LEANDRO, CA.
ALL RIGHTS RESERVED WORLDWIDE
MADE IN U.S.A.

TOLERANCES UNLESS OTHERWISE SPECIFIED FRACTIONS ± DEC. ± ANGLES ±		SIO CABLE A ASSEMBLY	
APPROVALS	DATE		91-0400003
DRAWN YP	8/76		
CHECKED DZH	8/11/76	SCALE —— SIZE B	DRAWING NO. 91-2040003
		DO NOT SCALE DRAWING	SHEET 1 OF 1

8080 DEVELOPMENT SYSTEM FIRMWARE

This EROM memory board allows the user to edit, write, save, move, dump, and assemble 8080 programs written in standard 8080 mnemonics. It contains eight 5204 EROMs, which must be plugged into like numbered sockets on the board (note matching numbers on top side of EROM ICs). There are also 8 empty sockets for user implemented firmware and/or extensions and modifications on the present firmware.

F000

MEMORY REQUIREMENTS

1) The EROM board itself must be configured in the last page of memroy, or page 7.
2) There must be at least 2000 bytes of RAM memory, starting with location D000, available to the system firmware. D000 through D0FF comprise temporary storage and stack; D100 through D7FF contain the symbol table.
3) Please note the above requirements do not include memory allocation for user files.

I/O REQUIREMENTS

The software addresses three I/O devices as follows:

DEVICE 0 Teletype (UART)---status register. Bit 6 = 1 indicates data is available from the keyboard; i.e. a key has been struck. Bit 7 = 1 indicates that the teleprinter is ready to print another character.

DEVICE 1 Input is the ASCII input from the keyboard. Output is the ASCII output to the teleprinter.

DEVICE FF (Applies to Altair 8800 sense switches) sense switch 7 is to be used in conjunction with the list command, to be discussed in detail later.

OVERVIEW OF COMMANDS

GODBOUT

NUMB AAAA

CONTROL X	One of two panic buttons (see below)
DUMP	Display memory data
PAGE	Move a page of data
ENTR	Enter data to memory
EXEC	Execute a command
FILE	Create, assign, or display file information
NNNN	Any 4 numeric digits enter the editor (where N = decimal integer)
LIST	List file
DELT	Delete lines of a file
ASSM	Assemble a source file to object code

The executive has an error message, WHAT?, to indicate and improper command or an error in parameters following the command. WARNING: It is possible to confuse the executive with improper use of spaces. It is important to follow the command format exactly as detailed in the following sections; in examples of format, the symbol ¤ will be used to designate a space.

PANIC BUTTONS

1) A RUB-OUT deletes the last character typed in the line, if there is one; otherwise it does nothing.
2) CONTROL X deletes the entire line.

NOTE: No other non-printing commands are recognized by the executive.

COMMANDS

DUMP COMMAND This command examines portions of memory, with the values expressed in hexadecimal.

DUMP COMMAND FORMAT DUMP¤lower address¤upper address CR. The upper address may be omitted, in which case only the lower memory location is displayed.

EXAMPLE OF DUMP COMMAND

```
DUMP 1Ø1Ø 1Ø2A CR
1Ø1Ø 52 41 4D 2Ø 46 4F 52 2Ø 43 52 41 49 47 ØD Ø8 3Ø
1Ø2Ø 3Ø 3Ø 31 2Ø 2A ØD 14 3Ø 3Ø 31 3Ø
```

(Example 1)

PAGE COMMAND This command moves a page of data.

PAGE COMMAND FORMAT PAGE¤AA¤BB CR. This command relocates page AA to page BB, where AA and BB are two digit hexadecimal numbers, each of which designates one of 256 byte "pages".

ENTER COMMAND This command alters the contents of memory.
ENTER COMMAND FORMAT ENTR¤hexadecimal address CR. Software responds with a line feed; user may now enter as many two digit hex numbers as desired. These numbers must be separated by at least one space and/or carriage return if there are too many numbers to fit comfortably on a single line. A slash (/) followed by a carriage return terminates the command. Illegal digits (i.e. non-hex or more than two digits) will not be accepted.
EXAMPLE OF ENTER COMMAND

```
DUMP 2400 241F
2400 21 22 23 03 00 80 01 00 00 80 07 00 80 10 01 81
2410 01 80 84 10 81 90 81 94 41 00 01 40 00 40 80 40        (before)

ENTR 2400
01 25 34 67 8A 00 04                                        (enter
56 76 9F/                                                    command)

DUMP 2400 241F
2400 01 25 34 67 8A 00 04 56 76 9F 07 00 80 10 01 81        (after)
2410 01 80 84 10 81 90 81 94 41 00 01 40 00 40 80 40        (Example 2)
```

EXECUTE COMMAND Use this command to leave the Software Development Firmware and start execution of another program.
EXECUTE COMMAND FORMAT EXEC¤hex address CR

ILE COMMANDS These commands accomplish 4 separate file operations. These operations llow the user to create, delete, or activate files, and also to list file information.

CREATE FILE COMMAND Use this command to create files.
CREATE FILE COMMAND FORMAT FILE¤/FILE-NAME/¤hex address CR. The spaces are important! FILE-NAME should be six or fewer characters. "Hex address" denotes starting address of the file; the ending address of the file is dynamic and varies with the amount of information contained in the file. The executive responds to this command by typing a line feed and carriage return, followed by FILE-NAME, then typing "hex address" twice. The user may create up to six files; if he attempts to create the same file twice, system responds with NO¤NO.

DELETE FILE COMMAND Use this command to delete files.
DELETE FILE COMMAND FORMAT Use the following kluge: FILE¤/OLD-FILE-NAME/¤Ø CR.
(This takes advantage of the convention that no files are allowed to have a starting address of Ø.)

LISTING FILES COMMAND Use this command to find out which files you have floating around.
LISTING FILES COMMAND FORMAT FILES CR. This produces a listing of current files, with upper and lower bound address information.
LISTING FILES COMMAND EXAMPLE

```
        FILES
        TESTØ 1ØØØ  1Ø5D
        CRAG  2ØØØ  2ØØØ
        TEST1 14ØØ  14ØØ
        JUMK  24ØØ  24ØØ                                    (Example 3)
```

ACTIVATE FILE COMMAND This command specifies which file the file access commands should reference.
ACTIVATE FILE COMMAND FORMAT FILE¤/FILE-NAME/ CR. The system responds as above by

printing "FILE-NAME" plus parameters. Until another activate command, any file access command will reference this file. (For information on file access commands, see below). NOTE: To list the file (plus parameters) that is currently being referenced by the file ccess commands, enter FILE CR.

DITOR COMMANDS

NNNN COMMAND The purpose of this command is to insert line-oriented information into files. To use NNNN requires an active file---through creating a new file, having already activated a file, or by activating an existing file (see file commands above).

NNNN COMMAND FORMAT The user enters exactly four decimal digits, which means leading zeroes must be appended to any number under 1,000. The information from the first numeral, up to and including the carriage return, is inserted into the current file upon striking the carriage return. The line's position in the file is determined solely by the first four digits. It is advisable to leave gaps in the numerical sequence of the lines to allow for later line insertions without necessitating renumbering of the original lines. Existing lines may be altered by typing the number of that line, followed by the new information and a CR. Upon receipt of the CR, the systen deletes the old line and enters the new line in the same place.

LIST COMMAND Use this command to list files.

LIST COMMAND FORMAT LIST¤decimal number. This causes the system to type at least one line from the currently active file starting at line N. Upon completion of typing the present line, the system checks sense switch 8. If sense switch 8 is up, the command is terminated. If sense switch 8 is down, the next sequential line is printed if there is one. Encountering the last line of the current file unconditionally terminates the command.

DELETE COMMAND Use this command to delete lines of a file.

LETE COMMAND FORMAT DELT¤L1¤L2 CR. L1 and L2 are decimal line numbers; L2 should either larger than L1 or missing. If L2 is missing, exactly one line designated by .. is deleted from the current file. If L2 is present, all lines between L1 and L2, including both L1 and L2, are deleted.

ASSEMBLER

ASSEMBLE COMMAND Use this command to assemble a source program written by the user and located in the file area (see full description below)

ASSEMBLE COMMAND FORMAT ASSM¤(E)¤AAAA¤BBBB The assembler performs the assembly, assigning addresses to the object code starting at AAAA, where AAAA is a hexadecimal address. On the second pass the object code is placed in memory starting at location BBBB, where BBBB is again a hexadecimal address. If BBBB is not specified it assumes the same value as AAAA. During pass 1 certain errors are displayed and during pass 2 a complete listing is produced. If the optional E is specified in the command, only those lines which contain errors are listed.

ASSEMBLER DETAILED DESCRIPTION The assembler, once given control by the executive, proceeds to translate a symbolic 8080 assembly language (source) program into 8080 machine (object) code. To use the assembler, it is necessary to have an active file---this comes about by creating a new file, having already activated a file, or by activating an existing file (see file commands above). The assembler is a two pass assembler with the following features:

---Free format source input
---Symbolic addressing, including forward references and relative symbolic references
---Complex expressions may be used as arguments
---Self-defining constants
---Multiple constant forms
---Up to 256 character symbols
---Reserved names for 8080 registers
---ASCII character code generation
---6 pseudo operations (assembler directives)

GODBOUT

The assembler translates those lines contained in the current file into object code. Since the second character following the line number is considered to be the first source code character position, it should normally be blank (see STATEMENTS section for exceptions). Line numbers are reproduced on the listing, but are not processed by the assembler.

The assembler assembles a source program composed of STATEMENTS, COMMENTS, and PSEUDO-OPERATIONS.

During pass 1, the assembler 1) allocates all storage necessary for the translated program and 2) creates a symbol table and defines the values of all symbols used. The storage allocated for the object code begins at the first byte dictated by the first parameter in the original Executive ASSM Command.

During pass 2, all expressions, symbols, and ASCII constants are evaluated to their absolute values, and are placed in allocated memory in the appropriate locations; simultaneously, the listing indicates exactly what data is in each location of memory.

STATEMENTS

These may contain either symbolic 8080 machine instructions or pseudo-ops.
STATEMENT FORMAT - LABEL/NAME OPERATION OPERAND COMMENT
The NAME FIELD, if present must begin in assembler character position 1. The symbol in the name field can contain as many characters as desired; however, the symbol table only uses the first 5 characters to uniquely define a symbol. All symbols in this field must begin with an alphabetic character and may contain no special characters (such as #, ¢, etc.)
The OPERATION FIELD contains either an 8080 operation mnemonic or a system pseudo-operation code.
The OPERAND FIELD contains parameters that pertain to the operation in the operation eld. If two arguments are present they must be separated by a comma.
e COMMENT FIELD is for explanatory remarks, and is listed without processing. Lines wnich consist entirely of comments must begin with an asterisk (*) in character position 1.
All fields must be separated one from the other by the presence of one or more spaces.
EXAMPLE:

```
0033 CLIP  MOV M,B  COMMENT
0045 * COMMENT
0078       JMP  ARF
0095       CALL CLIP
0100 ARF   ADI  8+2-6
0140       MOV  A,B
```

(Example 4)

SYMBOLIC NAMES

To assign a symbolic label to a statement, merely place the symbol in the NAME FIELD. To omit the name field, skip two or more spaces after the line number and begin the operation field. If a label is attached to a statement, the assembler assigns it the value of the current Location Counter. The Location Counter always holds the address of the next byte to be assembled. The only exception to this is the EQU pseudo-op, wherein a symbol in the NAME FIELD is assigned a value which is contained in the OPERAND FIELD of the EQU pseudo-op statement.
EXAMPLE:

```
0057  GEORG EQU 166
```

(Example 5)

assigns the value 166 to the name GEORG. Thus, anywhere in the program GEORG has the value 166.

LABELS are defined when they appear in the NAME FIELD. All defined labels may be used as mbolic arguments in the ARGUMENT FIELD (see example 4).

In addition to user defined names, the assembler has reserved several symbols with predetermined values. These names may not be used except in the OPERAND FIELD. They are: (value in parenthesis)

A - the accumulator (7)
B - register B (0)
C - register C (1)
D - register D (2)
E - register E (3)
H - register H (4)
L - register L (5)
M - memory (through H, L) (6)

In addition to the above reserved symbols, there is a single special character symbol ($). This symbol changes in value as the assembly progresses, and is always equated with the value of the program counter after the current instruction is assembled. It may only be used in the OPERAND FIELD. EXAMPLE:

```
        JMP $       means jump to the next location after this instruction; i.e. the
        MOV B,H     MOV instruction

        LDA $+2
        DB  0
        DB  1       means load the data at the second location after this location.  In
        DB  2       this case the data has the value 2.
        DB  3
        DB  4
        DB  5                                                          (Example 5)
```

RELATIVE SYMBOLIC ADDRESSING

If the name of a particular location is known, a nearby location may be specified using the known name and a numeric offset. EXAMPLE:

```
        JMP   LBL
        JPE   LBL+4
        CC    SUB
        CALL  $+1AH
    LBL MOV   B,C
        STC   B,C
        MVI   C, 'B'
        INR   M                                                        (Example 6)
```

In this example the instruction LBL refers to the MOV B,C instruction. The instruction JPE LBL+4 refers to the INR M instruction. LBL+4 means the address LBL plus four bytes. This form of addressing can be used to locate several bytes before or after a named location.

CONSTANTS

The assembler allows the user to write positive or negative numbers directly in a statement. Numbers are regarded as decimal constants and their binary equivalents will be used appropriately; all unsigned numbers are assumed positive. Decimal constants may be defined using the descriptor "D" after the numeric value; constants without descriptors are assumed to be decimal.

Hexadecimal constants may be defined using the descriptor "H" after a numeric value; however, a hexadecimal constant cannot start with the digits A-F, and must include a leading Ø. This enables the assembler to differentiate between a numeric value and a symbol.

ASCII constants may be defined by enclosing the ASCII character within single quote marks, i.e. 'C'. For double word constants two characters may be defined within one quote string.

EXPRESSIONS

An expression is a sequence of one or more symbols, constants, or other expressions, separated by the arithmetic operators plus or minus, EXAMPLE:

```
        NIT +4
        GORG-'A'+73
        GUNF+38H-7                                                     (Example 7)
```

Expressions are calculated using 16 bit arithmetic. All arithmetic is done modulo 65536. Single byte data cannot contain a value greater than or equal to 127 or less than -128. Any value outside this range will result in an assembler error.

PSEUDO-OPERATIONS

The pseudo-operations are written as ordinary statements, but they direct the assembler to perform certain functions which do not always develop 8080 machine code. The following describes the pseudo-ops.

ORG---Set Program Origin.

FORMAT: Label ORG expression

"Label" is optional, but if present will be equated to the above expression.

END---End of Assembly: this pseudo-op informs the assembler that the last source statement has been read. The assembler will then start on pass 2 or terminate the assembly and pass control back to the executive. This pseudo-op is not needed when assembling from a memory file since the assembler will stop when an end of file indicator has been reached.

EQU---Equate Symbolic Value

FORMAT: Label EQU expression, where "label" is a symbol the value of which will be determined from the expression; "expression" is an expression which when evaluated will be assigned to the symbol given in the name field.

DS---Define Storage: The DS causes the assembler to advance the Assembly Program Counter, effectively skipping past a given number of memory bytes.

FORMAT: Label DS expression

DB---Define Byte: This pseudo-op is used to reserve one byte of storage. The content of the byte is specified in the argument field.

FORMAT: Label DB expression

DW---Define Word: This pseudo-op is used to define two bytes of storage. The evaluated argument will be placed in the two bytes; high order 8 bits in the low order byte, and the low order 8 bits in the high order byte. This conforms to the Intel format for two byte expressions.

ASSEMBLER ERRORS

The following error flags are output on the assembler listing when the error occurs. Some of the errors are only output during pass 1.

O	Opcode error
L	Label error
D	Duplicate label error
M	Missing label error
V	Value error
U	Undefined symbol
S	Syntax error
R	Register error
A	Argument error

GODBOUT
BILL GODBOUT ELECTRONICS
BOX 2355, OAKLAND AIRPORT, CA 94614

GODBOUT


```
        THIS ROUTINE INITIALIZES THE FILE AREA FOR SUBSEQUENT PROCESSING
              ORG     H'FOOO'
ITA           LXI     H,FILEO                    21 ØØ DØ
              MVI     C,MAXFIL*FELEN
              XRA     A
INIT2         MOV     M,A
              INX     H
              DCR     C
              JNZ     INIT2

        THIS IS THE STARTING POINT OF THE SELF CONTAINED SYSTEM ONCE
        THE SYSTEM HAS BEEN INITIALIZED.  COMMANDS ARE READ FROM THE USER,
        EXECUTED AND CONTROL RETURNS BACK TO THIS POINT TO READ ANOTHER
        USER COMMAND.

SYSB          LXI     SP,AREA+18
              CALL    READ                       READ INPUT LINE
              INX     H
              MOV     A,M                        FETCH FIRST CHARACTER
              CPI     A'9'+1                     COMMAND OR LINE NUMBER
              JC      LINE                       JUMP IF LINE FOR FILE
              CALL    VALC                       GET COMMAND VALUES
              CALL    COMM                       CHECK LEGAL COMMANDS
EOR           CALL    CRLF                       GET HERE WHEN ROUTINE IS DONE
              JMP     SYSB

        THIS ROUTINE READS IN A LINE FROM THE TTY AND PLACES IT IN AN
        INPUT BUFFER
        THE FOLLOWING ARE SPECIAL CHARACTERS
          CR    -TERMINATES READ ROUTINE
          LF    -NOT RECOGNIZED BY ROUTINE
          CONTROL X DELETE CURRENT LINE
          DEL   -DELETE CHARACTER
        ALL DISPLAYABLE CHARACTERS BETWEEN BLANK-Z AND THE ABOVE
        ARE RECOGNIZED BY THE READ ROUTINE ALL OTHER ARE SKIPPED
        OVER.  THE ROUTINE WILL NOT ACCEPT MORE CHARACTERS THAN THE INPUT
        BUFFER WILL HOLD

READ          LXI     H,IBUF                     SET INPUT BUFFER ADDRESS
              SHLD    ADDS                       SAVE ADDRESS
              MVI     E,2                        INITIALIZE CHARACTER COUNT
NEXT          CALL    IN8                        READ A LINE
              MOV     A,B
              CPI     24                         CHECK FOR CONTROL X
              JNZ     CR
              CALL    CRLF                       OUTPUT A CRLF
              JMP     READ
CR            CPI     ASCR                       GET A ASCII CR
              JNZ     DEL
              MOV     A,L
              CPI     >IBUF                      CHECK FOR FIRST CHARACTER
              JZ      READ
              MVI     M,ASCR                     PLACE CR AT END OF LINE
              INX     H
              MVI     M,1                        PLACE EOF INDICATOR IN LINE
              INX     H
              MVI     A,>IBUF+83
              CALL    CLER                       CLEAR REMAINING BUFFER
              LXI     H,IBUF-1
              MOV     M,E                        SAVE CHARACTER COUNT
              RET
DEL           CPI     127                        CHECK FOR DELETE CHARACTER
              JNZ     CHAR
              MVI     A,>IBUF
              CMP     L                          IS THIS 1ST CHARACTER
              JZ      NEXT
              DCX     H                          DECREMENT POINTER
              DCR     E                          DECREMENT COUNT
BSPA          MVI     B,H'5F'
              CALL    OUT8
              JMP     NEXT
CHAR          CPI     A' '                       CHECK FOR LEGAL CHARACTERS
              JC      NEXT
              CPI     A'Z'+1
              JNC     NEXT
              MOV     B,A
              CALL    OUT8                       ECHO CHARACTER
              MOV     M,A
              MVI     A,>IBUF+81
              CMP     L                          CHECK FOR END OF LINE
              JZ      BSPA
              INX     H
              INR     E                          INCREMENT CHARACTER COUNT
              JMP     NEXT

        THIS ROUTINE IS USED TO BLANK OUT A PORTION OF MEMORY

CLER          CMP     L
              RZ
              MVI     M,A'                       PLACE BLANK IN MEMORY
              INX     H
              JMP     CLER

        THIS SUBROUTINE IS USED TO READ A BYTE OF DATA FROM THE UART

IN8           IN      USTA                       READ UART STATUS
              ANI     H'40'
```



```
              JZ      IN8
              IN      UDAI                       READ DATA
              ANI     127                        STRIP OFF PARITY
              MOV     B,A
              RET

        THIS ROUTINE OUTPUTS A BYTE OF DATA TO THE UART

OUT8          IN      USTA                       READ STATUS
              ANI     H'80'
              JZ      OUT8
OK            MOV     A,B
              OUT     UDAO                       TRANSMIT DATA
              RET

        THIS ROUTINE WILL OUTPUT A CARRIAGE RETURN AND LINE FEED.

CRLF          MVI     B,13                       CR
              CALL    OUT8
LF            MVI     B,10                       LF
              CALL    OUT8
              RET

        THIS ROUTINE JUMPS TO A LOCATION IN MEMORY GIVEN BY THE
        INPUT COMMAND AND BEGINS EXECUTION OF PROGRAM THERE.

EXEC          CALL    VCHK                       CHECK FOR PARAMETER
              CALL    CRLF
              LHLD    BBUF                       FETCH ADDRESS
              PCHL                               JUMP TO PROGRAM

        THIS ROUTINE CHECK THE INPUT COMMAND AGAINST ALL LEGAL COMMANDS
        STORED IN A TABLE.  IF A LEGAL COMMAND IS FOUND A JUMP IS
        MADE TO THAT ROUTINE. OTHERWISE AN ERROR MESSAGE IS OUTPUT
         ) THE USER

              LXI     D,CTAB                     COMMAND TABLE ADDRESS
              MVI     B,NCOM                     NUMBER OF COMMANDS
              MVI     A,4                        LENGTH OF COMMAND
              STA     NCHR                       SAVE
              CALL    COMS                       SEARCH TABLE
              JNZ     WHAT                       JUMP IF ILLEGAL COMMAND
              PCHL                               JUMP TO ROUTINE
```



```
        THIS ROUTINE CHECKS IF A BASE CHARACTER STRING IS EQUAL TO
        ANY OF THE STRINGS CONTAINED IN A TABLE POINTED TO BY
        D,E.  THE LENGTH OF THE STRINGS ARE <256.  ON RETURN IF THE
        ZERO FLAG IS SET A MATCH WAS FOUND, IF THE ZERO FLAG IS CLEAR (0)
        NO MATCH WAS FOUND.  REGISTER B CONTAINS THE NUMBER OF
        STRINGS TO COMPARE
        THE TABLE CONSISTS OF ANY NUMBER OF CHAR. WITH 2 BYTES CONTAINING
        VALUES ASSOCIATED WITH IT.  IT CAN BE USED TO SEARCH THROUGH
        A COMMAND TABLE OR SYMBOL TABLE.
        ON RETURN D,E POINT TO THE LAST BYTE ASSOCIATED WITH THE
        CHARACTER STRING IF A MATCH WAS FOUND.  IF NO MATCH WAS
        FOUND D,E POINT TO THE NEXT LOCATION AFTER THE END OF THE TABLE

COMS          LHLD    ADDS                       FETCH COMPARE ADDRESS
              LDA     NCHR                       GET LENGTH OF STRING
              MOV     C,A
              CALL    SEAR                       COMPARE STRINGS
              LDAX    D                          FETCH VALUE
              MOV     H,A
              INX     D
              LDAX    D                          FETCH VALUE
              MOV     L,A
              RZ
              INX     D                          SET TO NEXT STRING
              DCR     B                          DECREMENT COUNT
              JNZ     COMS
              INR     B                          CLEAR ZERO FLAG
              RET

        THIS ROUTINE CHECKS IF TWO CHARACTER STRINGS CONTAINED IN MEMORY
        ARE EQUAL.  THE STRINGS ARE POINTED TO BY H,L AND D,E.
        ON RETURN THE ZERO FLAG SET INDICATES A MATCH.  REGISTER C
        INDICATES THE LENGTH OF THE STRINGS.  ON RETURN THE POINTERS
        POINT TO THE NEXT ADDRESS AFTER THE CHARACTER STRINGS

SEAR          LDAX    D                          FETCH CHARACTER
              CMP     M                          COMPARE CHARACTERS
              JNZ     INCA
              INX     H
              INX     D
              DCR     C                          DECREMENT CHARACTER COUNT
              JNZ     SEAR
              RET
INCA          INX     D
              DCR     C
              JNZ     INCA
              INR     C                          CLEAR ZERO FLAG
              RET
```

```
        THIS ROUTINE ZEROS OUT A BUFFER IN MEMORY WHICH IS THEN
        USED BY OTHER SCANNING ROUTINES

ZBUF    XRA     A               GET A ZERO
        LXI     D,ABUF+12       BUFFER ADDRESS
        MVI     B,12            BUFFER LENGTH
ZBU1    DCX     D               DECREMENT ADDRESS
        STAX    D               ZERO BUFFER
        DCR     B
        JNZ     ZBU1
        RET

        THIS ROUTINE CALL ETRA TO OBTAIN THE INPUT PARAMETER VALUES
        AND CALLS AN ERROR ROUTINE IF AN ERROR OCCURRED IN THAT ROUTINE

VALC    CALL    ETRA            GET INPUT PARAMETERS
        JC      WHAT            JUMP IF ERROR
        RET

        THIS ROUTINE EXTRACTS THE VALUES ASSOCIATED WITH A COMMAND
        FROM THE INPUT STREAM AND PLACES THEM IN THE ASCII BUFFER (ABUF).
        IT ALSO CALLS A ROUTINE TO CONVERT THE ASCII HEXADECIMAL TO BINARY
        AND STORES THEM IN THE BINARY BUFFER (BBUF)
        ON RETURN CARRY SET INDICATES AN ERROR IN INPUT PARAMETERS

ETRA    LXI     H,0             GET A ZERO
        SHLD    BBUF+2          ZERO VALUE
        SHLD    FBUF            SET NO FILE NAME
        CALL    ZBUF            ZERO BUFFER
        LXI     H,IBUF-1        INPUT BUFFER ADDRESS
VAL1    INX     H
        MOV     A,M             FETCH INPUT CHARACTER
        CPI     A' '            LOOK FOR FIRST BLANK
        CMC
        RNC                     RETURN IF CARRY
        JNZ     VAL1            JUMP IF NO BLANK
        SHLD    PNTR            SAVE POINTER
        CALL    SBLK            SCAN TO FIRST PARAMETER
        CMC
        RNC                     RETURN IF CR
        CPI     A':'
        JNZ     VAL5            NO FILE NAME
        LXI     D,FBUF          NAME FOLLOWS PUT IN FBUF
        MVI     C,NMLEN
VAL2    INX     H
        MOV     A,M
        CPI     A'/'
        JZ      VAL3
        DCR     C
        JM      WHAT
        STAX    D               STORE FILE NAME
        NOP
        INX     D
        JMP     VAL2
VAL3    MVI     A,A'            GET AN ASCII SPACE
VAL4    DCR     C
        JM      DONE
        STAX    D               FILL IN WITH SPACES
        INX     D
        JMP     VAL4
DONE    CALL    SBL2
        CMC
        RNC
VAL5    LXI     D,ABUF
        CALL    ALPS            PLACE PARAMETER IN BUFFER
        MOV     A,B             GET DIGIT COUNT
        CPI     5               CHECK NUMBER OF DIGITS
        CMC
        RC                      RETURN IF TOO MANY DIGITS
        LXI     B,ABUF
        CALL    AHEX            CONVERT VALUE
        RC                      ILLEGAL CHARACTER
        SHLD    BBUF            SAVE IN BINARY BUFFER
        LXI     H,ABUF
        CALL    NORM            NORMALIZE ASCII VALUE
        CALL    SBLK            SCAN TO NEXT PARAMETER
        CMC
        RNC                     RETURN IF CR
        LXI     D,ABUF+4
        CALL    ALPS            PLACE PARAMETER IN BUFFER
        MOV     A,B             GET DIGIT COUNT
        CPI     5               CHECK NUMBER OF DIGITS
        CMC
        RC                      RETURN IF TOO MANY DIGITS
        LXI     B,ABUF+4
        CALL    AHEX            CONVERT VALUE
        RC                      ILLEGAL CHARACTER
        SHLD    BBUF+2          SAVE IN BINARY BUFFER
        LXI     H,ABUF+4
        CALL    NORM            NORMALIZE ASCII VALUE
        ORA     A               CLEAR CARRY
        RET

        THIS ROUTINE FETCHES DIGITS FROM THE BUFFER ADDRESSED BY
        REGISTERS B,C AND CONVERTS AND ASCII DECIMAL DIGITS INTO
        BINARY.  UP TO A 16 BIT VALUE CAN BE CONVERTED.  THE SCAN
        STOPS WHEN A BINARY ZERO IS FOUND IN THE BUFFER

ADEC    LXI     H,0             GET A 16 BIT ZERO
ADE1    LDAX    B               FETCH ASCII DIGIT
        ORA     A               SET ZERO FLAG
        RZ                      RETURN IF FINISHED
        MOV     D,H             SAVE CURRENT VALUE
        MOV     E,L             SAVE CURRENT VALUE
        DAD     H               TIMES TWO
        DAD     H               TIMES TWO
        DAD     D               ADD IN ORIGINAL VALUE
        DAD     H               TIMES TWO
        SUI     48              ASCII BIAS
        CPI     10              CHECK FOR LEGAL VALUE
        CMC
        RC                      RETURN IF ERROR
        MOV     E,A
        MVI     D,0
        DAD     D               ADD IN NEXT DIGIT
        INX     B               INCREMENT POINTER
        JMP     ADE1

        THIS ROUTINE FETCHES DIGITS FROM THE BUFFER ADDRESSED BY
        REGISTERS B,C AND CONVERTS THE ASCII HEXADECIMAL DIGITS INTO
        BINARY.  UP TO A 16 BIT VALUE CAN BE CONVERTED.  THE SCAN STOPS
        WHEN A BINARY ZERO IS FOUND IN THE BUFFER

AHEX    LXI     H,0             GET A 16 BIT ZERO
AHE1    LDAX    B               FETCH ASCII DIGIT
        ORA     A               SET ZERO FLAG
        RZ                      RETURN IF ZERO
        DAD     H               LEFT SHIFT
        DAD     H               LEFT SHIFT
        DAD     H               LEFT SHIFT
        DAD     H               LEFT SHIFT
        CALL    AHS1            CONVERT TO BINARY
        CPI     H'10'           CHECK FOR LEGAL VALUW
        CMC
        RC                      RETURN IF ERROR
        ADD     L
        MOV     L,A
        INX     B               INCREMENT POINTER
        JMP     AHE1

        THIS SUBROUTINE CONVERTS ASCII HEX DIGITS INTO BINARY

AHS1    SUI     48              ASCII BIAS
        CPI     10              DIGIT 0-10
        RC
        SUI     7               ALPHA BIAS
        RET

        THIS ROUTINE CONVERTS A BINARY VALUE TO ASCII HEXADECIMAL
        AND OUTPUTS THE CHARACTERS TO THE TTY

HOUT    CALL    BINH            CONVERT VALUE
        LXI     H,HCON          CONVERSION AREA
CHOT    MOV     B,M             FETCH OUTPUT CHARACTER
        CALL    OUT8            OUTPUT CHARACTER
        INX     H
        MOV     B,M             FETCH CHARACTER
        CALL    OUT8            OUTPUT CHARACTER
        RET

        THIS ROUTINE DOES THE SAME AS ABOVE BUT OUTPUTS A BLANK
        AFTER THE LAST CHARACTER

HOTB    CALL    HOUT            CONVERT AND OUTPUT
        CALL    BLK1            OUTPUT A BLANK

        THIS ROUTINE CONVERTS A BINARY VALUE TO ASCII DECIMAL
        DIGITS AND OUTPUTS THE CHARACTERS TO THE TTY

DOUT    CALL    BIND            CONVERT VALUE
        CALL    HOUT+3          OUTPUT VALUE (2 DIGITS)
        INX     H
        MOV     B,M             GET LAST DIGIT
        CALL    OUT8            OUTPUT
        RET

        THIS ROUTINE OUTPUTS A BLANK

BLK1    MVI     B,A' '          GET A BLANK
        CALL    OUT8
        RET

        THIS ROUTINE IS USED BY OTHER ROUTINES TO INCREMENT THE
        STARTING ADDRESS IN A COMMAND AND COMPARE IT WITH THE FINAL
        ADDRESS IN THE COMMAND.  ON RETURN THE CARRY FLAG SET
        INDICATES THAT THE FINAL ADDRESS HAS BEEN REACHED

ACHK    LHLD    BBUF            FETCH START ADDRESS
        LDA     BBUF+3          STOP ADDRESS (HIGH)
        CMP     H               COMPARE ADDRESSES
        JNZ     ACH1
        LDA     BBUF+2          STOP ADDRESS (LOW)
        CMP     L               COMPARE ADDRESSES
        JNZ     ACH1
        STC                     SET CARRY IF EQUAL
ACH1    INX     H               INCREMENT START ADDRESS
        SHLD    BBUF            STORE START ADDRESS
        RET

        THIS ROUTINE OUTPUTS CHARACTERS FROM A CHARACTER STRING UNTIL
        A CARRIAGE RETURN IS FOUND

SCRN    MOV     B,M             FETCH CHARACTER
        MVI     A,13            CARRIAGE RETURN
        CMP     B               CHARACTER=CR
        RZ
        CALL    OUT8            OUTPUT CHARACTER
        INX     H               INCREMENT ADDRESS
        JMP     SCRN
```

```
   THIS ROUTINE CONVERTS THE BINARY VALUE IN REG A INTO
   ASCII HEXADECIMAL DIGITS AND STORES THEM IN MEMORY

         LXI      H,HCON          CONVERSION ADDRESS
         MOV      B,A             SAVE VALUE
         RAR
         RAR
         RAR
         RAR
         CALL     BIN1
         MOV      M,A
         INX      H
         MOV      A,B
         CALL     BIN1            CONVERT TO ASCII
         MOV      M,A
         RET

   THIS ROUTINE CONVERTS A VALUE TO HEXADECIMAL

BIN1     ANI      H'0F'           LOW FOUR BITS
         ADI      48              MODIFY FOR ASCII
         CPI      58              DIGIT 0-9
         RC
         ADI      7               MODIFY FOR A-F
         RET

   THIS ROUTINE CONVERTS THE BINARY VALUE IN REG A INTO
   ASCII DECIMAL DIGITS AND STORES THEM IN MEMORY

BIND     LXI      H,HCON          CONVERSION ADDRESS
         MVI      B,100
         CALL     BID1            CONVERT HUNDREDS DIGIT
         MVI      B,10
         CALL     BID1            CONVERT TENS DIGIT
         ADI      A'0'            GET UNITS DIGIT
         MOV      M,A             STORE IN MEMORY
         RET

   THIS ROUTINE CONVERTS A VALUE TO DECIMAL

BID1     MVI      M,A'0'-1        INITIALIZE DIGIT COUNT
         INR      M
         SUB      B               CHECK DIGIT
         JNC      BID1+2
         ADD      B               RESTORE VALUE
         INX      H
         RET


   LEGAL COMMAND TABLE
```



```
CTAB     DATA     A'DUMP'         DUMP COMMAND
         DDB      DUMP            COMMAND ADDRESS
         DATA     A'EXEC'         EXECUTE COMMAND
         DDB      EXEC            COMMAND ADDRESS
         DATA     A'ENTR'         ENTER COMMAND
         DDB      ENTR            COMMAND ADDRESS
         DATA     A'FILE'         FILE COMMAND
         DDB      FILE            COMMAND ADDRESS
         DATA     A'LIST'         LIST COMMAND
         DDB      LIST            COMMAND ADDRESS
         DATA     A'DELT'         DELETE COMMAND
         DDB      DELL            COMMAND ADDRESS
         DATA     A'ASSM'         ASSEMBLE COMMAND
         DDB      ASSM            COMMAND ADDRESS
         DATA     A'PAGE'         PAGE TRANSFER COMMAND
         DDB      PAGE            COMMAND ADDRESS
         DATA     A'PROM'         PROM PROGRAM COMMAND
         DDB      PROM            DEFINE COMMAND ADDRESS
         DATA     A'CUST'         CUST COMMAND
         DDB      H'E000'         COMMAND ADDRESS

   THIS ROUTINE CHECKS IF ANY PARAMETERS WERE ENTERED
   WITH THE COMMAND, IF NOT AN ERROR MESSAGE IS ISSUED

VCHK     LDA      ABUF            FETCH PARAMETER BYTE
         ORA      A               SET FLAGS
         JZ       WHAT            NO PARAMETER
         RET

   THIS ROUTINE DUMPS OUT THE CONTENTS OF MEMORY FROM
   THE START TO FINAL ADDRESSES GIVEN IN THE COMMAND

DUMP     CALL     VCHK            CHECK FOR PARAMETERS
         MVI      A,16            LOCATIONS PER LINE
         STA      SCNT            DUMP COUNTER
DUMS     CALL     CRLF            START NEW LINE
         LDA      BBUF+1          FETCH ADDRESS
         CALL     HOUT            OUTPUT ADDRESS
         LDA      BBUF
         CALL     HOTB            OUTPUT ADDRESS
         LDA      SCNT            FETCH LINE COUNTER
DUM1     STA      DCNT
         LHLD     BBUF            FETCH MEMORY ADDRESS
         MOV      A,L             GET LOW ORDER ADDRESS
         OUT      PADO            SET PROM ADDRESS
         MOV      A,M
         CPI      H'EF'
         MOV      A,M
         JNZ      DUM2
         IN       PDAI            READ PROM DATA
DUM2     CALL     HOTB            OUTPUT VALUE
         CALL     ACHK            CHECK ADDRESS
```



```
         RC                       RETURN IF FINISHED
         LDA      DCNT            FETCH COUNTER
         DCR      A               DECREMENT COUNTER
         JNZ      DUM1
         JMP      DUMS

   THIS ROUTINE WILL MOVE 1 PAGE (256 BYTES) FROM 1ST ADDRESS GIVEN
   IN COMMAND TO 2ND ADDRESS IN COMMAND

PAGE     CALL     VCHK            CHECK FOR PARAMETER
         LDA      ABUF+4          FETCH 2ND PARAMETER
         ORA      A               DOES 2ND PARAMETER EXIST
         JZ       WHAT
         LHLD     BBUF            FETCH MOVE FROM ADDRESS
         XCHG
         LHLD     BBUF+2          FETCH MOVE TO ADDRESS
         MVI      B,0             SET COUNTER
PAG1     MOV      A,E
         OUT      PADO            SET PROM ADDRESS
         MOV      A,D
         CPI      H'FF'           CHECK FOR PROM ADDRESS
         LDAX     D               GET DATA
         JNZ      PAG2
         IN       PDAI            READ PROM DATA
PAG2     MOV      M,A
         INX      H
         INX      D
         DCR      B               DECREMENT COUNT
         JNZ      PAG1
         RET

   THIS ROUTINE INITIALIZES THE BEGINNING OF FILE ADDRESS
   AND END OF FILE ADDRESS AS WELL AS INITIALIZES THE FILE AREA
   WHEN THE FILE COMMAND IS USED

         ORA      A
FILE     CALL     CRLF
   CHECK FOR FILE PARAMETERS
         LDA      FBUF
         ORA      A
         JZ       FOUT            NO - GO LIST
         CALL     FSEA            LOOK UP FILE
         XCHG                     PNTR IN DE
         JNZ      TEST            IF FOUND
      ENTRY
         LDA      ABUF            CHECK FOR PARAM
         ORA      A
         JZ       WHA1            NO?? - GIVE EM HELL
   CHECK FOR ROOM IN DIRECTORY
         LDA      FEF
         ORA      A
         JNZ      ROOM
```



```
         LXI      H,EMES1
         JMP      MESS
   ENTRY FOUND ARE THESE PARAMETERS
TEST     LDA      ABUF
         ORA      A
         JZ       SWAPS
         LHLD     BBUF
         MOV      A,H
         ORA      L
         JZ       SWAPS
         LXI      H,EMES2         NO-NO CAN'T DO
         JMP      MESS            IT - DELETE FIRST
   MOVE FILE NAME TO BLOCK POINTED TO BY FREAD
ROOM     LHLD     FREAD           DIRECT POINTER IN D,E
         XCHG                     FILE NAME POINTER IN H,L
         LXI      H,FBUF
         PUSH     D
         MVI      C,NMLEN         NAME LENGTH COUNT
MOV23    MOV      A,M
         STAX     D
         INX      D
         DCR      C               TEST COUNT
         JMP      PATCH 81
P+3      POP      D               RESTORE ENTRY PTR, MAKE CURRENT
   MAKE FILE POINTED TO BY D,E CURRENT
SWAPS    LXI      H,FILE0
         MVI      C,FELEN         ENTRY LENGTH
SWAP     LDAX     D
         MOV      B,M             EXCHANGE
         MOV      M,A
         MOV      A,B
         STAX     D
         INX      D               BUMP POINTERS
         INX      H
         DCR      C               TEST COUNT
         JNZ      SWAP
   CHECK FOR 2ND PARAMETER, = >INITIALIZE NEW
         LDA      ABUF
         ORA      A
         JZ       FOOT            NO SECOND PARAMETER
   PROCESS 2ND PARAMETER
         LHLD     BBUF            GET ADDRESS
         SHLD     BOFP            SET BEGIN
         SHLD     EOFP            SET END
         MOV      A,L             IS ADDRESS ZERO
         ORA      H
         JZ       FIL35           YES
FIL30    MVI      M,1             NON ZERO - SET EOF
FIL35    XRA      A
         STA      MAXL            AND MAX LINE 0
         JMP      FOOT            OUTPUT PARAMETERS
FOUT     LDA      IBUF+4
         CPI      A'S'            IS COMMAND FILES
         MVI      C,MAXFIL
```

```
         JZ        FOUL
FOOT     MVI       C,1
   OUTPUT THE # OF ENTRIES IN C
FOUL     LXI       H,FILE0
         MOV       A,C
   NE    STA       FOCNT          SAVE COUNT
         PUSH      H
         LXI       D,NMLEN
         DAD       D
         MOV       A,M
         ORA       A
         JNZ       FOOD           NONZERO, OK TO OUTPUT
         INX       H
         ADD       M
         INX       H
         JNZ       FOOD
         INX       SP
         INX       SP
         INX       H
         INX       H
         JMP       FEET
   HAVE AN ENTRY TO OUTPUT
FOOD     POP       H              PTR
         MVI       C,NMLEN
FAST     MOV       B,M            LOAD CHARACTER TO B
         CALL      OUTB           OUTPUT
         DCR       C
         INX       H
         JNZ       FAST           DO THE REST
   NOW OUTPUT BEGIN-END PTRS
         CALL      FOOL           OUTPUT BEGIN
         CALL      FOOL           OUTPUT END
         CALL      CRLF           AND C/R
   TEST COUNT, H,L POINTS PAST EOFP
FEET     LXI       D,FELEN-NMLEN-4
         DAD       D              MOVE TO NEXT ENTRY
         LDA       FOCNT
         DCR       A              TEST COUNT
         JNZ       FINE           MORE TO DO
         RET                      DONE!!!
   OUTPUT NUMBER POINTED TO BY H,L
   ON RET, H,L POINT 2 WORDS LATER
FOOL     CALL      BLK1           SPACE
         INX       H
         MOV       A,M
         DCX       H
         PUSH      H
         CALL      HOUT           OUTPUT
         POP       H
         MOV       A,M
         INX       H
         INX       H
         PUSH      H
         CALL      HOTB           OUTPUT
```



```
         POP       H              RESTORE H,L
         RET
   SEARCH THE FILE DIRECTORY FOR THE FILE
   WHOSE NAME IS IN FBUF.
   RETURN IF FOUND, ZERO IS OFF, H,L POINT TO
   ENTRY WHILE SEARCHING.  ON ENTRY FOUND WITH ADDR
   ZERO, SET FEF TO >0 AND FREAD TO THE ADDR OF ENTRY.

FSEA     XRA       A
         STA       FEF            CLAIM NO FREE ENTRIES
         MVI       B,MAXFIL       COUNT OF ENTRIES
         LXI       D,FILE0        TABLE ADDRESS
FSE10    LXI       H,FBUF
         MVI       C,NMLEN
         CALL      SEAR           TEST STRINGS
         PUSH      PSW            SAVE FLAG
         PUSH      D
         LDAX      D              GET BOFP
         ORA       A               EMPTY ENTRY?
         JNZ       FSE20
         INX       D              TEST OTHER WORD
         LDAX      D
         ORA       A
         JNZ       FSE20          NOPE-GO TEST FOR MATCH
         XCHG                     H,L GET MIDDLE OF FREE ENTRY
         LXI       D,-NMLEN-1
         DAD       D              MOVE TO BEGINNING
         SHLD      FREAD          SAVE ADDR
         MOV       A,D
         STA       FEF            SET FREE ENTRY FOUND
         POP       H              RESTORE INTERIM PTR
         POP       PSW            UNJUNK STACK
   MOVE TO NEXT ENTRY
FSE15    LXI       D,FELEN-NMLEN
         DAD       D
         XCHG                     NEXT ENTRY ADDR IN DE
         DCR       B              TEST COUNT
         RZ                       DONE---NOPE
         JMP       FSE10          TRY NEXT
   ENTRY WASN'T FREE, TEST FOR MATCH
FSE20    POP       H
         POP       PSW
         JNZ       FSE15          IF ZERO CLEAR, NO MATCH
   ENTRY FOUND
         LXI       D,-NMLEN       BACKUP
         DAD       D              H,L POINTS TO ENTRY
         MOV       A,D
         ORA       A              CLEAR ZERO
         RET                      THATS ALL

   OUTPUT ERROR MESSAGE FOR ILLEGAL COMMAND

WHAT     CALL      CRLF           OUT CRLF
```



```
WHA1     LXI       H,EMES         MESSAGE ADDRESS
MESS     CALL      SCRN
         JMP       EOR

EMES     DATA      A'WHAT?'

         DATA      13             CARRIAGE RETURN
EMES1    DATA      A'FULL',13

EMES2    DATA      A'NO NO',13


   CALL ROUTINE TO ENTER DATA INTO MEMORY
   AND CHECK FOR ERROR ON RETURN

   THIS ROUTINE IS USED TO ENTER DATA VALUES INTO MEMORY.
   EACH VALUE IS ONE BYTE AND IS WRITTEN IN HEXADECIMAL
   VALUES GREATER THAN 255 WILL CAUSE CARRY TO BE SET
   AND RETURN MADE TO CALLING PROGRAM

ENTR     CALL      VCHK           CHECK  FOR PARAMETERS
         CALL      ENTS
         JC        WHAT
         CALL      CRLF
         RET

EEND     EQU       A'/'           TERMINATION CHARACTER
ENTS     CALL      CRLF
         CALL      READ           READ INPUT DATA
         LXI       H,IBUF         SET LINE POINTER
         SHLD      PNTR           SAVE POINTER
ENT1     CALL      ZBUF           CLEAR BUFFER
         CALL      SBLK           SCAN TO FIRST VALUE
         JC        ENTS           JUMP IF CR FOUND
         CPI       EEND
         RZ                       RETURN (CARRY IS ZERO)
         CALL      ALPS           PLACE VALUE IN BUFFER
         MOV       A,B            GET DIGIT COUNT
         CPI       3              CHECK NUMBER OF DIGITS
         CMC
         RC                       RETURN IF MORE THAN 2 DIGITS
         LXI       B,ABUF         CONVERSION ADDRESS
         CALL      AHEX           CONVERT VALUE
         RC                       ERROR IN HEX CHARACTER
         MOV       A,L
         LHLD      BBUF           FETCH MEMORY ADDRESS
         MOV       M,A            PUT IN MEMORY
         CALL      ACM1           INCREMENT MEMORY LOCATION
         JMP       ENT1
```



```
   THIS ROUTINE IS USED TO ENTER LINES INTO THE FILE
   AREA.  THE LINE NUMBER IS FIRST CHECKED TO SEE IF IT IS
   A VALID NUMBER (0000-9999). NEXT IT IS CHECKED TO SEE IF IT IS
   GREATER THAN THE MAXIMUM CURRENT LINE NUMBER.  IF IT IS THE NEW
   LINE IS INSERTED AT THE END OF THE CURRENT FILE AND THE MAXIMUM
   LINE NUMBER IS UPDATED AS WELL AS THE END OF FILE POSITION
   LINE NUMBERS THAT ALREADY EXIST ARE INSERTED INTO THE FILE AREA
   AT THE APPROPRIATE PLACE AND ANY EXTRA CHARACTERS IN THE OLD
   LINE ARE DELETED

LINE     MVI       C,4            NO OF DIGITS TO CHECK
         LXI       H,IBUF-1       INITIALIZE ADDRESS
LICK     INX       H
         MOV       A,M            FETCH LINE DIGIT
         CPI       A'0'           CHECK FOR VALID NUMBER
         JC        WHAT
         CPI       A'9'+1
         JNC       WHAT
         DCR       C
         JNZ       LICK
         SHLD      ADDS           FIND ADDRESS
         LXI       D,MAXL+3       SET ADDRESS
         CALL      COMO
         JNC       INSR
   GET HERE IF NEW LINE IS GREATER THAN MAXIMUM LINE
         INX       H
         CALL      LODM           GET NEW LINE NUMBER
         LXI       H,MAXL+3
         CALL      STOM           MAKE IT MAXIMUM LINE NUMBER
         LXI       D,IBUF-1
         LHLD      EOFP           END OF FILE POSITION
         MVI       C,1
         CALL      LMOV           PLACE LINE IN FILE
SEOF     MOV       M,1            END OF FILE INDICATOR
         SHLD      EOFP           END OF FILE ADDRESS
         JMP       EOR
   GET HERE IF NEW LINE MUST BY INSERTED INTO ALREADY EXISTING
   FILE AREA
INSR     CALL      FIN1           FIND LINE IN FILE
         MVI       C,2
         JZ        EQUL           NEW LN NOT EQUAL TO SOME OLD LN
         DCR       C
EQUL     MOV       B,M
         DCX       H
         MVI       M,2            MOVE LINE INDICATOR
         SHLD      INSP           INSERT LINE POSITION
         LDA       IBUF-1         NEW LN COUNT
         DCR       C
         JZ        LT             NEW LN NOT = OLD LN
         SUB       B              COUNT DIFFERENCE
         JZ        ZERO           LINE LENGTHS EQUAL
         JC        GT
   GET HERE IF NO OF CHARS IN OLD LINE ) NO OF CHARS IN NEW LINE
   OR NEW LINE NUMBER WAS NOT EQUAL TO SOME OLD LINE NUMBER
```

```
LT      LHLD    EOFP            END OF FILE ADDRESS
        MOV     D,H
        MOV     E,L
        CALL    ADR             MOVE TO ADDRESS
        SHLD    EOFP            NEW END FO FILE ADDRESS
        MVI     C,2
        CALL    RMOV            OPEN UP FILE AREA
        JMP     ZERO
   GET HERE IF NO OF CHARS IN OLD LINE, NO OF CHARS IN NEW LINE
_T      CMA
        INR     A               COUNT DIFFERENCE
        MOV     D,H
        MOV     E,L
        CALL    ADR
        XCHG
        CALL    LMOV            DELETE EXCESS CHARACTERS IN FILE
        MVI     M,1             E-O-F INDICATOR
        SHLD    EOFP            E-O-F ADDRESS
   GET HERE TO INSERT CURRENT LINE INTO FILE AREA
ZERO    LHLD    INSP            INSERT ADDRESS
        MVI     M,ASCR
        INX     H
        LXI     D,IBUF-1        NEW LINE ADDRESS
        MVI     C,1             CHECK VALUE
        CALL    LMOV            PLACE LINE IN FILE
        JMP     EOR
```

THIS ROUTINE IS USED TO FIND A LN IN THE FILE AREA
WHICH IS GREATER THAN OR EQUAL TO THE CURRENT LINE NUMBER

```
FIND    LXI     H,ABUF+3        BUFFER ADDRESS
        SHLD    ADDS            SAVE ADDRESS
FIN1    LHLD    BOFP            BEGIN FILE ADDRESS
FI1     CALL    EO1             CHECK FOR END OF FILE
        XCHG
        LHLD    ADDS            FETCH FIND ADDRESS
        XCHG
        MVI     A,4
        CALL    ADR             LN ADDRESS
        CALL    COM0            COMPARE LINE NUMBERS
        RC
        RZ
F12     MOV     A,M
        CALL    ADR             NEXT LINE ADDRESS
        JMP     FI1
```

THIS ROUTINE CHECKS IF THE CURRENT ADDRESS
WHEN SEARCHING THROUGH THE FILE AREA IS THE END OF FILE

```
EOF     INX     H
EO1     MVI     A,1             E-O-F INDICATOR
        CMP     M
```



```
        RNZ
        JMP     EOR
```

THIS ROUTINE IS USED TO ADD A VALUE TO AN ADDRESS
CONTAINED IN REGISTER H,L

```
ADR     ADD     L
        MOV     L,A
        RNC
        INR     H
        RET
```

THIS ROUTINE WILL MOVE CHARACTER STRINGS FROM ONE LOCATION
OF MEMORY TO ANOTHER
CHARACTERS ARE MOVED FROM LOCATION ADDRESSED BY D,E TO LOCATION
ADDRESSED BY H,L. ADDITIONAL CHARACTERS ARE MOVED BY
INCREMENTING MEMORY UNTIL THE CHARACTER IN REGISTER C IS FETCHED

```
LMOV    LDAX    D               FETCH CHARACTER
        INX     D               INCREMENT FETCH ADDRESS
        CMP     C               TERMINATION CHARACTER
        RZ
        MOV     M,A             STORE CHARACTER
        INX     H               INCREMENT STORE ADDRESS
        JMP     LMOV
```

THIS ROUTINE IS SIMILAR TO ABOVE EXCEPT THAT THE CHARACTER ADDRESS
IS DECREMENTED AFTER EACH FETCH AND STORE

```
RMOV    LDAX    D               FETCH CHARACTER
        DCX     D               DECREMENT FETCH ADDRESS
        CMP     C               TERMINATION CHARACTER
        RZ
        MOV     M,A             STORE CHARACTER
        DCX     H               DECREMENT STORE ADDRESS
        JMP     RMOV
```

THIS ROUTINE IS USED TO LOAD FOUR CHARACTERS FROM
MEMORY INTO REGISTERS

```
LODM    MOV     B,M             FETCH CHARACTER
        INX     H
        MOV     C,M             FETCH CHARACTER
        INX     H
        MOV     D,M             FETCH CHARACTER
        INX     H
        MOV     E,M             FETCH CHARACTER
        RET
```


THIS ROUTINE STORES FOUR CHARACTERS FROM REGISTERS
INTO MEMORY

```
STOM    MOV     M,E             STORE CHARACTER
        DCX     H
        MOV     M,D             STORE CHARACTER
        DCX     H
        MOV     M,C             STORE CHARACTER
        DCX     H
        MOV     M,B             STORE CHARACTER
        RET
```

THIS ROUTINE IS USED TO COMPARE TWO CHARACTER STRINGS
OF LENGTH 4. ON RETURN ZERO FLAG SET MEANS BOTH
STRINGS ARE EQUAL. CARRY FLAG = 0 MEANS STRING ADDRESSED
BY D,E WAS GREATER THAN OR EQUAL TO CHARACTER STRING
ADDRESSED BY H,L

```
COM0    MVI     B,1             EQUAL COUNTER
        MVI     C,4             STRING LENGTH
        ORA     A               CLEAR CARRY
CO1     LDAX    D               FETCH CHARACTER
        SBB     M               COMPARE CHARACTERS
        JZ      CO2
        INR     B               INCREMENT EQUAL COUNTER
CO2     DCX     D
        DCX     H
        DCR     C
        JNZ     CO1
        DCR     B
        RET
```

THIS ROUTINE IS SIMILAR TO THE ABOVE ROUTINE EXCEPT ON
RETURN CARRY FLAG = 0 MEANS THAT CHARACTER STRING ADDRESSED
BY D,E IS ONLY GREATER THAN STRING ADDRESSED BY H,L

```
COM1    MVI     C,4             STRING LENGTH
        LDAX    D               FETCH CHARACTER
        SUI     1
        JMP     CO1+1
```

THIS ROUTINE WILL TAKE ASCII CHARACTERS AND ADD ANY
NECESSARY ASCII ZEROS SO THE RESULT IS A 4 CHARACTER
ASCII VALUE

```
NORM    CALL    LODM            LOAD CHARACTERS
        XRA     A               FETCH A ZERO
        CMP     B
        RZ
NOR1    CMP     E
        CNZ     STOM            STORE VALUES
        RNZ
```



```
        MOV     E,D             NORMALIZE VALUE
        MOV     D,C
        MOV     C,B
        MVI     D,A'0'
        JMP     NOR1
```

THIS ROUTINE IS USED TO LIST THE CONTENTS OF THE FILE
AREA STARTING AT THE LINE NUMBER GIVEN IN THE COMMAND

```
LIST    CALL    CRLF
        CALL    FIND            FIND STARTING LN
        INX     H
LIS1    CALL    SCRN            OUTPUT LINE
        CALL    CRLF
        CALL    EOF             END OF FILE
        IN      SWCH            READ SWITCHES
        ANI     H'80'
        RNZ
        INX     H
        JMP     LIS1
```

THIS ROUTINE IS USED TO PROGRAM A 1702A PROM

```
PROM    CALL    VCHK            CHECK FOR PARAMETER
PRO1    CALL    CRLF
        LDA     BBUF            GET ADDRESS (LOW)
        CALL    HOTB            OUTPUT ADDRESS
        MVI     D,3             NUMBER OF ATTEMPTS
        LHLD    BBUF            GET ADDRESS
PRO2    MOV     A,L
        OUT     PADO
        MOV     A,M             GET DATA
        OUT     PDAO            OUTPUT TO PROM
        MVI     A,2
        OUT     PCTO            ENABLE PROGRAMMER
        CALL    DLAY            500 MSEC DELAY
        XRA     A               GET A ZERO
        OUT     PCTO            DISABLE PROGRAMMER
        IN      PDAI            READ DATA
        CMP     M               COMPARE DATA
        JZ      PRO3
        MVI     B,A'?'          ERROR INDICATOR
        CALL    OUTB
        DCR     D               NUMBER OF ATTEMPTS
        JNZ     PRO2            TRY AGAIN
        RET
PRO3    CALL    ACHK            FINAL ADDRESS
        RC                      RETURN IF FINISHED
        JMP     PRO1            NEXT LOCATION

DLAY    MVI     E,150
DLA1    XRA     A               GET A ZERO (256)
DLA2    DCR     A
```

```
        JNZ     DLA7
        DCR     E
        JNZ     DLA1
        RET
```

THIS ROUTINE IS USED TO DELETE LINES FROM THE FILE AREA
THE REMAINING FILE AREA IS THEN MOVED IN MEMORY SO THAT
THERE IS NO EXCESS SPACE IN MEMORY

```
DELL    CALL    VCHK            CHECK FOR PARAMETER
        CALL    FIND            FIND LINE IN FILE AREA
        SHLD    DELP            SAVE DELETE POSITION
        LXI     H,ABUF+7
        MOV     A,M             CHECK FOR 2ND PARAMETER
        ORA     A               SET FLAGS
        JNZ     DEL1
        LXI     H,ABUF+3        USE FIRST PARAMETER
DEL1    SHLD    ADDS            SAVE FIND ADDRESS
        XCHG
        LXI     H,MAXL+3
        CALL    COMD            COMPARE LINE NUMBERS
        LHLD    DELP            LOAD DELETE POSITION
        JC      NOVR
```

GET HERE IF DELETION INVOLVES END OF FILE

```
        SHLD    EOFP            CHANGE E-O-F
        MVI     M,1             SET E-O-F INDICATOR
        XCHG
        LHLD    BOFP            GET BEGIN FILE ADDRESS
        XCHG
        MVI     B,13            SET SCAN SWITCH
        DCX     H               DECREMENT FILE ADDRESS
DEL2    MOV     A,L             CHECK FOR BOF
        SUB     E
        MOV     A,H
        SBB     D
        MVI     A,ASCR          LOOK FOR CR
        JC      DEL4            DECREMENTED PAST BOF
        DCR     B
        DCX     H               FIND NEW MAX LN
        CMP     M
        JNZ     DEL2
        DCX     H
        MOV     A,L
        SUB     E
        MOV     A,H
        SBB     D
        JC      DEL5
        CMP     M               END OF PREVIOUS LINE
        INX     H
        INX     H
        JZ      DEL3
        INX     H
DEL3    CALL    LODM            LOAD NEW MAX LN
```



```
        LXI     H,MAXL+3        SET ADDRESS
        CALL    STUM            STORE NEW MAX LN
        RET
DEL4    CMP     B               CHECK SWITCH
DEL5    XCHG
        JNZ     DEL3-1
        STA     MAXL            MAKE MAX LN A SMALL NUMBER
        RET
```

GET HERE IF DELETION IS IN MIDDLE OF FILE AREA

```
NOVR    CALL    F11             FIND END OF DELETE AREA
        CZ      F12             NEXT LINE IF THIS LN IS EQUAL
NOV1    XCHG
        LHLD    DELP            CHAR MOVE TO POSITION
        MVI     C,1             MOVE TERMINATOR
        CALL    LMOV            COMPACT FILE AREA
        SHLD    EOFP            SET EOF POSITION
        MVI     M,1             SET EOF INDICATOR
        RET
```

STARTING HERE IS THE SELF ASSEMBLER PROGREM
THIS PROGRAM ASSEMBLER PROGRAMS WHICH ARE
IN THE FILE AREA

```
ASSM    CALL    VCHK            CHECK FOR PARAMETER
        LDA     ABUF+4          GET 2ND PARAMETER
        ORA     A               CHECK FOR PARAMETERS
        JNZ     ASM4
        LHLD    BBUF            FETCH 1ST PARAMETER
        SHLD    BBUF+2          STORE INTO 2ND PARAMETER
ASM4    LDA     IBUF+4          FETCH INPUT CHARACTER
        CPI     A'E'            ERROR ONLY INDICATOR
        JNZ     ASM5
        XRA     A               SET FOR ONLY ERRORS
ASM5    STA     AERR            SET ERROR SWITCH
        ERA     A               GET A ZERO
        STA     NOLA            INITIALIZE LABEL COUNT
ASM3    STA     PAS1            SET PASS INDICATOR
        LHLD    BBUF            FETCH ORIGIN
        SHLD    ASPC            INITIALIZE PC
        LHLD    BOFP            GET START OF FILE
        SHLD    APNT            SAVE ADDRESS
ASM1    LHLD    APNT            FETCH LINE POINTER
        LXI     SP,AREA+18
        MOV     A,M             FETCH CHARACTER
        CPI     1               END OF FILE
        JZ      EASS            JUMP IF END OF FILE
        XCHG
        INX     D               INCREMENT ADDRESS
        LXI     H,OBUF          BLANK START ADDRESS
        MVI     A,>IBUF-5       BLANK END ADDRESS
        CALL    CLER            BLANK OUT BUFFER
        MVI     C,ASCR          STOP CHARACTER
        CALL    LMOV            MOVE LINE INTO BUFFER
```



```
        MOV     M,C             PLACE CR IN BUFFER
        XCHG
        SHLD    APNT            SAVE ADDRESS
        LDA     PAS1            FETCH PASS INDICATOR
        ORA     A               SET FLAGS
        JNZ     ASM2            JUMP IF PASS 2
        CALL    PAS1
        JMP     ASM1
ASM2    CALL    PAS2
        LXI     H,OBUF          OUTPUT BUFFER ADDRESS
        CALL    AOUT            OUTPUT LINE
        JMP     ASM1
```

THIS ROUTINE IS USED TO OUTPUT THE LISTING FOR AN ASSEMBLY
IT CHECKS WHETHER ALL LINES ARE PRINTED OR ONLY THOSE
WITH ERRORS DEPENDING UPON THE ERROR SWITCH

```
AOUT    LDA     AERR            FETCH ERROR SWITCH
        ORA     A               SET FLAGS
        JNZ     AOU1            OUTPUT ALL LINES
AOU2    LDA     OBUF+18         FETCH ERROR INDICATOR
        CPI     A' '            CHECK FOR AN ERROR
        RZ                      RETURN IF NO ERROR
AOU1    LXI     H,OBUF          OUTPUT BUFFER ADDRESS
        CALL    CRLF
        CALL    SCRN            OUTPUT LINE
        RET
```

PASS 1 OF ASSEMBLER. USED TO FORM SYMBOL TABLE

```
PAS1    CALL    ZBUF            CLEAR BUFFER
        STA     PAS1            SET FOR PASS 1
        LXI     H,IBUF          INITIALIZE LINE POINTER
        SHLD    PNTR            SAVE ADDRESS
        MOV     A,M             FETCH CHARACTER
        CPI     A' '            CHECK FOR A BLANK
        JZ      OPC             JUMP IF NO LABEL
        CPI     A'*'            CHECK FOR COMMENT
        RZ                      RETURN IF COMMENT
```

PROCESS LABEL

```
        CALL    SLAB            GET AND CHECK LABEL
        JC      OPS             ERROR IN LABEL
        JZ      ERRD            DUPLICATE LABEL
        CALL    LCHK            CHECK CHARACTER AFTER LABEL
        JNZ     OPS             ERROR IF NO BLANK
        MVI     C,LLAB          LENGTH OF LABELS
        LXI     H,ABUF          SET BUFFER ADDRESS
MLAB    MOV     A,M             FETCH NEXT CHARACTER
        STAX    D               STORE IN SYMBOL TABLE
        INX     D
        INX     H
        DCR     C               DECREMENT COUNT
```



```
        JNZ     MLAB
        XCHG
        SHLD    TABA            SAVE TABLE ADDRESS FOR EQU
        LDA     ASPC+1          FETCH PC (HIGH)
        MOV     M,A             STORE IN TABLE
        INX     H
        LDA     ASPC            FETCH PC (LOW)
        MOV     M,A             STORE IN TABLE
        LXI     H,NOLA
        INR     M               INCREMENT NUMBER OF LABELS
```

PROCESS OPCODE

```
OPC     CALL    ZBUF            ZERO WORKING BUFFER
        CALL    SBLK            SCAN TO OPCODE
        JC      OERR            FOUND CARRIAGE RETURN
        CALL    ALPS            PLACE OPCODE IN BUFFER
        CPI     A' '            CHECK FOR BLANK AFTER OPCODE
        JC      OPCD            CR AFTER OPCODE
        JNZ     OERR            ERROR IF NO BLANK
        JMP     OPCD            CHECK OPCODE
```

THIS ROUTINE CHECKS THE CHARACTER AFTER A LABEL FOR A BLANK
OR A COLON

```
LCHK    LHLD    PNTR
        MOV     A,M             GET CHARACTER AFTER LABEL
        CPI     A' '            CHECK FOR A BLANK
        RZ                      RETURN IF A BLANK
        CPI     A':'            CHECK FOR A COLON
        RNZ
        INX     H
        SHLD    PNTR            SAVE POINTER
        RET
```

PROCESS ANY PSEUDO OPS THAT NEED TO BE IN PASS 1

```
PSU1    CALL    SBLK            SCAN TO OPERAND
        LDAX    D               FETCH VALUE
        ORA     A               SET FLAGS
        JZ      ORG1            ORG OPCODE
        JM      DAT1            DATA STATEMENT
        JPO     EQU1            EQU OPCODE
        CPI     5
        JC      RES1            RES OPCODE
        JNZ     EASS            JUMP IF END
```

DO DW PSEUDO/OP

```
ACO1    MVI     C,2             2 BYTE INSTRUCTION
        XRA     A               GET A ZERO
        JMP     OCN1            ADD VALUE TO PROGRAM COUNTER
```

DO ORG PSEUDO-OP

```
ORG1    CALL    ASCN            GET OPERAND
        LDA     OBUF+18         FETCH ERROR INDICATOR
        CPI     A' '            CHECK FOR AN ERROR
        RNZ                     IF ERROR DON'T CHANGE PC
```

```
         SHLD    ASPC        STORE NEW ORIGIN
         LDA     IBUF        GET FIRST CHARACTER
         CPI     A' '        CHECK FOR LABEL
         RZ                  NO LABEL
         JMP     EQUS        CHANGE LABEL VALUE
   DO EQU PSEUDO-OP
EQU1     CALL    ASCN        GET OPERAND
         LDA     IBUF        FETCH LST CHARACTER
         CPI     A' '        CHECK FOR LABEL
         JZ      ERRM        MISSING LABEL
EQUS     XCHG
         LHLD    TABA        SYMBOL TABLE ADDRESS
         MOV     M,D         STORE LABEL VALUE
         INX     H
         MOV     M,E
         JMP     AOU2        OUTPUT IF ERROR
   DO DS PSEUDO-OP
RES1     CALL    ASCN        GET OPERAND
         MOV     B,H
         MOV     C,L
         JMP     RE21        ADD VALUE TO PROGRAM COUNTER

   DO DB PSEUDO-OP
DAT1     JMP     OP2

   PERFORM PASS 2 OF THE ASSEMBLER

PAS2     LXI     H,OBUF      SET OUTPUT BUFFER ADDRESS
         LDA     ASPC+1      FETCH PC (HIGH)
         CALL    BINH+3      CONVERT FOR OUTPUT
         INX     H
         LDA     ASPC        FETCH PC (LOW)
         CALL    BINH+3      CONVERT FOR OUTPUT
         SHLD    OIND        SAVE OUTPUT ADDRESS
         CALL    ZBUF        CLEAR BUFFER
         LXI     H,IBUF      INITIALIZE LINE POINTER
PABL     SHLD    PNTR        SAVE POINTER
         MOV     A,M         FETCH FIRST CHARACTER
         CPI     A' '        CHECK FOR LABEL
         JZ      OPC         GET OPCODE
         CPI     A'*'        CHECK FOR COMMENT
         RZ                  RETURN IF COMMENT
         CALL    SLAB        SCAN OFF LABEL
         JC      ERRL        ERROR IN LABEL
         CALL    LCHK        CHECK FOR A BLANK OR COLON
         JNZ     ERRL        ERROR IF NOT A BLANK
         JMP     OPC


   PROCESS PSEUDO OPS FOR PASS2
PSU2     LDAX    D
         ORA     A           SET FLAGS
         JZ      ORG2        ORG OPCODE
```



```
         JM      DAT2        DATA OPCODE
         RPO                 RETURN IF EQU
         CPI     5
         JC      RES2        RES OPCODE
         JNZ     EASS        END OPCODE
   DO DW PSEUDO-OP
ACO2     CALL    TYS6        GET VALUE
         JMP     ACO1
   DO DS PSEUDO-OP
RES2     CALL    ASBL        GET OPERAND
         MOV     B,M
         MOV     C,L
         LHLD    BBUF+2      FETCH STORAGE COUNTER
         DAD     B           ADD VALUE
         SHLD    BBUF+2      SAVE
RES21    XRA     A           GET A ZERO
         JMP     OCN2
   DO DB PSEUDO-OP
DAT2     CALL    TYS5        GET OPERAND
         XRA     A           GET A ZERO
         MVI     C,1         BYTE COUNT
         JMP     OCN1
   DO ORG PSEUDO-OP
ORG2     CALL    ASBL        GET NEW ORIGIN
         LDA     OBUF+18     GET ERROR INDICATOR
         CPI     A' '        CHECK FOR AN ERROR
         RNZ                 DON'T MODIFY PC IF ERROR
         XCHG
         LHLD    ASPC        FETCH PC
         XCHG
         SHLD    ASPC        STORE NEW PC
         MOV     A,L
         SUB     E           FORM DIFFERENCE OF ORIGINS
         MOV     E,A
         MOV     A,M
         SBB     D
         MOV     D,A
         LHLD    BBUF+2      FETCH STORAGE POINTER
         DAD     D           MODIFY
         SHLD    BBUF+2      SAVE
         RET

   PROCESS 1 BYTE INSTRUCTIONS WITHOUT OPERANDS

TYP1     CALL    ASTO        STORE VALUE IN MEMORY
         RET

   PROCESS STAX AND LDAX INSTRUCTIONS

TYP2     CALL    ASBL        FETCH OPERAND
         CNZ     ERRR        ILLEGAL REGISTER
         MOV     A,L         GET LOW ORDER OPERAND
         ORA     A           SET FLAGS
         JZ      TY31        OPERAND = 0
```



```
         CPI     2           OPERAND = 2
         CNZ     ERRR        ILLEGAL REGISTER
         JMP     TY31

   PROCESS PUSH,POP,INX,DCX,DAD INSTRUCTIONS

TYP3     CALL    ASBL        FETCH OPERAND
         CNZ     ERRR        ILLEGAL REGISTER
         MOV     A,L         GET LOW ORDER OPERAND
         RRC                 CHECK LOW ORDER BIT
         CC      ERRR        ILLEGAL REGISTER
         RAL                 RESTORE OPERAND
         CPI     8
         CNC     ERRR        ILLEGAL REGISTER
TY31     RLC                 MULTIPLY BY 8
         RAL
         RAL
TY32     MOV     B,A
         LDAX    D           FETCH OPCODE BASE
         ADD     B           FORM OPCODE
         CPI     118         CHECK FOR MOV M,M
         CZ      ERRR        ILLEGAL REGISTER
         JMP     TYP1

   PROCESS ACCUMULATOR, INR,DEC,MOV,RST INSTRUCTIONS

TYP4     CALL    ASBL        FETCH OPERAND
         CNZ     ERRR        ILLEGAL REGISTER
         MOV     A,L         GET LOW ORDER OPERAND
         CPI     8
         CNC     ERRR        ILLEGAL REGISTER
         LDAX    D           FETCH OPCODE BASE
         CPI     64          CHECK FOR MOV INSTRUCTION
         JZ      TY41
         CPI     199
         MOV     A,L
         JZ      TY31        RST INSTRUCTION
         JM      TY32        ACCUMULATOR INSTRUCTION
         JMP     TY31        INR,DCR
   PROCESS MOV INSTRUCTION
TY41     DAD     H           MULTIPLY OPERAND BY 8
         DAD     H
         DAD     H
         ADD     L           FORM OPCODE
         STAX    D           SAVE OPCODE
         CALL    MPNT        INCREMENT POINTER
         CALL    ASCN        GET NEXT OPERAND
         CNZ     ERRR        ILLEGAL REGISTER
         MOV     A,L         FETCH LOW ORDER OPERAND
         CPI     8
         CNC     ERRR        ILLEGAL REGISTER
         JMP     TY32

   PROCESS IMMEDIATE INSTRUCITONS
```



```
   IMMEDIATE BYTE CAN BE BETWEEN -128 AND +127
   MVI INSTRUCTION IS A SPECIAL CASE AND CONTAINS 2 ARGUMENTS
   IN OPERAND
TYP5     CPI     6           CHECK FOR MVI INSTRUCTION
         CZ      TY56
         CALL    ASTO        STORE OBJECT BYTE
TY55     CALL    ASBL        GET IMMEDIATE ARGUMENT
         INR     A
         CPI     2           CHECK OPERAND FOR RANGE
         CNC     ERRV        OPERAND OUT OF RANGE
         MOV     A,L
         JMP     TYP1

   FETCH 1ST ARGUMENT FOR MVI AND LXI INSTRUCTIONS

TY56     CALL    ASBL        FETCH ARGUMENT
         CNZ     ERRR        ILLEGAL REGISTER
         MOV     A,L         GET LOW ORDER ARGUMENT
         CPI     8
         CNC     ERRR        ILLEGAL REGISTER
         DAD     H           MULTIPLY BY 8
         DAD     H
         DAD     H

         LDAX    D           FETCH OPCODE BASE
         ADD     L           FORM OPCODE
         MOV     E,A         SAVE OBJECT BYTE
MPNT     LHLD    PNTR        FETCH POINTER
         MOV     A,M         FETCH CHARACTER
         CPI     A','        CHECK FOR COMMA
         INX     H           INCREMENT POINTER
         SHLD    PNTR
         JNZ     ERRS        SYNTAX ERROR IF NO COMMA
         MOV     A,E         GET OBJECT BYTE
         RET

   PROCESS 3 BYTE INSTRUCTIONS
   LXI INSTRUCTION IS A SPECIAL CASE

TYP6     CPI     1           CHECK FOR LXI INSTRUCTION
         JNZ     TY6         JUMP IF NOT LXI
         CALL    TY56        GET REGISTER
         ANI     H'08'       CHECK FOR ILLEGAL REGISTER
         CNZ     ERRR        REGISTER ERROR
         MOV     A,E         GET OPCODE
         ANI     H'F7'       CLEAR BIT IN ERROR
TY6      CALL    ASTO        STORE OBJECT BYTE
TYS6     CALL    ASBL        FETCH OPERAND
         MOV     A,L
         MOV     D,H         STORE 2ND BYTE
         CALL    ASTO
         MOV     A,D
         JMP     TYP1
         RET
```

```
   THIS ROUTINE IS USED TO STORE OBJECT CODE PRODUCED
   BY THE ASSEMBLER DURING PASS 2 INTO MEMORY

ASTO    LHLD    BBUF+2          FETCH STORAGE ADDRESS
        MOV     M,A             STORE OBJECT BYTE
        INX     H               INCREMENT LOCATION
        SHLD    BBUF+2
        LHLD    OIND            FETCH OUTPUT ADDRESS
        INX     H
        INX     H
        CALL    BINH+3          CONVERT OBJECT BYTE
        SHLD    OIND
        RET

   GET HERE WHEN END PSEUDO-OP IS FOUND OR WHEN END OF FILE
   OCCURS IN SOURCE STATEMENTS.  CONTROL IS SET FOR EITHER PASS 2
   OR ASSEMBLEY TERMINATES IF FINISHED.

EASS    LDA     PAS1            FETCH PASS INDICATOR
        ORA     A               SET FLAGS
        JNZ     EOR             JUMP IF FINISHED
        CALL    CRLF
        MVI     A,1             PASS INDICATOR FOR 2ND PASS
        JMP     ASM3            DO 2ND PASS


   THIS ROUTINE SCANS THROUGH A CHARACTER STRING UNTIL
   THE FIRST NON BLANK CHARACTER IS FOUND

   ON RETURN CARRY = 1 INDICATES A CR AS FIRST NON BLANK CHARACTER

SBLK    LHLD    PNTR            FETCH ADDRESS
SBL1    MOV     A,M             FETCH CHARACTER
        CPI     A' '            CHECK FOR A BLANK
        RNZ                     RETURN IF NON BLANK
SBL2    INX     H               INCREMENT
        SHLD    PNTR            SAVE POINTER
        JMP     SBL1


   THIS ROUTINE IS USED TO CHECK THE CONDITION CODE MNEMONICS
   FOR CONDITIONAL JUMPS, CALLS, AND RETURNS.

COND    LXI     H,ABUF+1
        SHLD    ADDS
        MVI     B,2             2 CHARACTERS
        CALL    COPC
        RET

   THE FOLLOWING IS THE OPCODE TABLE

OTAB    DATA    A'ORG'
```



```
        DATA    0
        DATA    0
        DATA    A'EQU'
        DATA    0
        DATA    1
        DATA    A'DB'
        DATA    0
        DATA    0
        DATA    -1
        DATA    A'DS'
        DATA    0
        DATA    0
        DATA    3
        DATA    A'DW'
        DATA    0
        DATA    0
        DATA    5
        DATA    A'END'
        DATA    0
        DATA    6
        DATA    0
        DATA    A'HLT'
        DATA    118
        DATA    A'RLC'
        DATA    7
        DATA    A'RRC'
        DATA    15
        DATA    A'RAL'
        DATA    23
        DATA    A'RAR'
        DATA    31
        DATA    A'RET'
        DATA    201
        DATA    A'CMA'
        DATA    47
        DATA    A'STC'
        DATA    55
        DATA    A'DAA'
        DATA    39
        DATA    A'CMC'
        DATA    63
        DATA    A'EI'
        DATA    0
        DATA    251
        DATA    A'DI'
        DATA    0
        DATA    243
        DATA    A'NOP'
        DATA    0
        DATA    0
        DATA    A'XCHG'
        DATA    235
        DATA    A'XTHL'
        DATA    227
```



```
        DATA    A'SPHL'
        DATA    249
        DATA    A'PCHL'
        DATA    223
        DATA    0
        DATA    A'STAX'
        DATA    2
        DATA    A'LDAX'
        DATA    10
        DATA    0
        DATA    A'PUSH'
        DATA    197
        DATA    A'POP'
        DATA    0
        DATA    193
        DATA    A'INX'
        DATA    0
        DATA    3
        DATA    A'DCX'
        DATA    0
        DATA    11
        DATA    A'DAD'
        DATA    0
        DATA    9
        DATA    0
        DATA    A'INR'
        DATA    4
        DATA    A'DCR'
        DATA    5
        DATA    A'MOV'
        DATA    64
        DATA    A'ADD'
        DATA    128
        DATA    A'ADC'
        DATA    136
        DATA    A'SUB'
        DATA    144
        DATA    A'SBB'
        DATA    152
        DATA    A'ANA'
        DATA    160
        DATA    A'XRA'
        DATA    168
        DATA    A'ORA'
        DATA    176
        DATA    A'CMP'
        DATA    184
        DATA    A'RST'
        DATA    199
        DATA    0
        DATA    A'ADI'
        DATA    198
        DATA    A'ACI'
        DATA    206
```



```
        DATA    A'SUI'
        DATA    214
        DATA    A'SBI'
        DATA    222
        DATA    A'ANI'
        DATA    230
        DATA    A'XRI'
        DATA    238
        DATA    A'ORI'
        DATA    246
        DATA    A'CPI'
        DATA    254
        DATA    A'IN'
        DATA    0
        DATA    219
        DATA    A'OUT'
        DATA    211
        DATA    A'MVI'
        DATA    6
        DATA    0
        DATA    A'JMP'
        DATA    0
        DATA    195
        DATA    A'CALL'
        DATA    205
        DATA    A'LXI'
        DATA    0
        DATA    1
        DATA    A'LDA'
        DATA    0
        DATA    58
        DATA    A'STA'
        DATA    0
        DATA    50
        DATA    A'SHLD
        DATA    34
        DATA    A'LHLD'
        DATA    42
        DATA    0
CONDITION TABLE
        DATA    A'NZ'
        DATA    0
        DATA    A'Z'
        DATA    0
        DATA    8
        DATA    A'NC'
        DATA    16
        DATA    A'C'
        DATA    0
        DATA    24
        DATA    A'PO'
        DATA    32
        DATA    A'PE'
        DATA    40
```

```
        DATA    A'P'
        DATA    0
        DATA    48
        DATA    A'M'
        DATA    0
        DATA    56
        DATA    0
```

THIS ROUTINE IS USED TO CHECK A GIVEN OPCOED AGAINST THE LEGAL OPCODES CONTAINED IN THE OPCODE TABLE

```
COPC    LHLD    ADDS
        LDAX    D               FETCH CHARACTER
        ORA     A               SET FLAGS
        JZ      COP1            JUMP IF TERMINATION CHARACTER
        MOV     C,B
        CALL    SEAR            COMPARE STRINGS
        LDAX    D
        RZ                      RETURN IF MATCH
        INX     D               NEXT STRING
        JMP     CPOC            CONTINUE SEARCH
COP1    INR     A               CLEAR ZERO FLAG
        INX     D               INCREMENT ADDRESS
        RET
```

THIS ROUTINE CHECKS THE LEGAL OPCODES IN BOTH PASS 1 AND PASS 2. IN PASS 1 THE PROGRAM COUNTER IS INCREMENTED BY THE CORRECT NUMBER OF BYTES. AN ADDRESS IS ALSO SET SO THAT AN INDEXED JUMP CAN BY MADE TO PROCESS THE OPCODE FOR PASS 2.

```
OPCD    LXI     H,ABUF          SET ADDRESS
        SHLD    ADDS
        LXI     D,OTAB          OPCODE TABLE ADDRESS
        MVI     B,4             CHARACTER COUNT
        CALL    COPC            CHECK OPCODES
        JZ      PSEU            JUMP IF A PSEUDO-OP
        DCR     B               3 CHARACTER OPCODES
        CALL    COPC
        JZ      OP1
        INR     B               4 CHAR OPCODES
        CALL    COPC
OP1     LXI     H,TYP1          TYPE 1 INSTRUCTIONS
OP2     MVI     C,1             1 BYTE INSTRUCTION
        JZ      OCNT

OPC2    CALL    COPC            CHECK FOR STAX,LDAX
        LXI     H,TYP2
        JZ      OP2
        CALL    COPC            CHECK FOR PUSH,POP,INX
                                DCX AND DAD
        LXI     H,TYP3
```



```
        JZ      OP2
        DCR     B               3 CHAR OPCODES
        CALL    COPC            ACCUMULATOR INSTRUCTIONS.
                                INR,DCR,MOV,RST
        LXI     H,TYP4
        JZ      OP2
OP3     CALL    COPC            IMMEDIATE INSTRUCTIONS
        LXI     H,TYP5
        MVI     C,2             2 BYTE INSTRUCTIONS
        JZ      OCNT
        INR     B               4 CHAR OPCODES
        CALL    COPC            JMP,CALL,LXI,LDA,STA,
                                LHLD,SHLD,OPCODES
        JZ      OP4
        CALL    COND            CONDITIONAL INSTRUCTIONS
        JNZ     OERR            ILLEGAL OPCODE
        ADI     192             ADD BASE VALUE OF RETURN
        MOV     D,A
        MVI     B,3             3 CHARACTER OPCODES
        LDA     ABUF            FETCH FIRST CHARACTER
        MOV     C,A             SAVE CHARACTER
        CPI     A'R'            CONDITIONAL RETURN
        MOV     A,D
        JZ      OP1
        MOV     A,C
        INR     D               FORM CONDITIONAL JUMP
        INR     D
        CPI     A'J'            CONDITIONAL JUMP
        JZ      OPAD
        CPI     A'C'            CONDITIONAL CALL
        JNZ     OERR            ILLEGAL OPCODE
        INR     D               FORM CONDITIONAL CALL
        INR     D
OPAD    MOV     A,D             GET OPCODE
OP4     LXI     H,TYP6
OP5     MVI     C,3             3 BYTE INSTRUCTION
OCNT    STA     TEMP            SAVE OPCODE
    CHECK FOR OPCODE ONLY CONTAINING THE CORRECT NUMBER OF CHARACTERS
    THUS SAY ADDQ WOULD GIVE AN ERROR
        MVI     A,>ABUF         LOAD BUFFER ADDRESS
        ADD     B               ADD LENGTH OF OPCODE
        MOV     E,A
        MVI     A,<ABUF         LOAD BUFFER ADDRESS
        ACI     0               GET HIGH ORDER ADDRESS
        MOV     D,A
        LDAX    D               FETCH CHARACTER AFTER OPCODE
        ORA     A               IT SHOULD BE ZERO
        JNZ     OERR            OPCODE ERROR
        LDA     PAS1            FETCH PASS INDICATOR
OCN1    MVI     B,0
        XCHG
OCN2    LHLD    ASPC            FETCH PROGRAM COUNTER
        DAD     B               ADD IN BYTE COUNT
```



```
        SHLD    ASPC            STORE PC
        ORA     A               WHICH PASS
        RZ                      RETURN IF PASS 1
        LDA     TEMP            FETCH OPCODE
        XCHG
        PCHL

OERR    LXI     H,ERRO          SET ERROR ADDRESS
        MVI     C,3             LEAVE 3 BYTES FOR PATCH
        JMP     OCN1-3

PSEU    LXI     H,ABUF+4        SET BUFFER ADDRESS
        MOV     A,M             FETCH CHARACTER AFTER OPCODE
        ORA     A               SHOULD BE A ZERO
        JNZ     OERR
        LDA     PAS1            FETCH PASS INDICATOR
        ORA     A
        JZ      PSU1
        JMP     PSU2
```

THIS ROUTINE IS USED TO PROCESS LABELS
IT CHECKS WHETHER A LABEL IS IN THE SYMBOL TABLE OR NOT
ON RETURN Z=1 MEANS A MATCH WAS FOUND AND H,L CONTAIN THE VALUE ASSOCIATED WITH THE LABEL. OTHERWISE D,E POINT TO THE NEXT AVAILABLE LOCATION IN THE TABLE. THE REGISTER NAMES A,B,C,D,E,H,L,M ARE PREDIFINED BY THE SYSTEM AND NEED NOT BE ENTERED BY THE USER
ON RETURN C=1 INDICATES A LABEL ERROR

```
SLAB    CPI     A'A'            CHECK FOR LEGAL CHARACTER
        RC                      RETURN IF LIIEGAL CHARACTER
        CPI     A'Z'+1          CHECK FOR ILLEGAL CHARACTER
        CMC
        RC                      RETURN IF ILLEGAL CHARACTER
        CALL    ALPS            PLACE SYMBOL IN BUFFER
        LXI     H,ABUF          SET BUFFER ADDRESS
        SHLD    ADDS            SAVE ADDRESS
        DCR     B               CHECK IF ONE CHARACTER
        JNZ     SLA1
    CHECK IF PREDEFINED REGISTER NAME
        INR     B               SET B=1
        LXI     D,RTAB          REGISTER TABLE ADDRESS
        CALL    COPC            CHECK NAME OF REGISTER
        JNZ     SLA1            NOT A PREDEFINED REGISTER
        MOV     L,A
        MVI     H,0             SET VALUE (HIGH)
        JMP     SLA2
SLA1    LDA     NOLA            FETCH SYMBOL COUNT
        MOV     B,A
        LXI     D,SYMT          SET SYMBOL TABLE ADDRESS
        ORA     A               ARE THERE ANY LABELS
        JZ      SLA3            JUMP IF NO LABELS
        MVI     A,LLAB          FETCH LENGTH OF LABEL
        STA     NCHR
        CALL    COMS            CHECK TABLE
```



```
SLA2    STC                     SET CARRY
        CMC                     CLEAR CARRY
        RET
SLA3    INR     A               CLEAR ZERO FLAG
        ORA     A               CLEAR CARRY
        RET
```

PREDEFINED REGISTER VALUES IN THIS TABLE

```
RTAB    DATA    A'A'
        DATA    7
        DATA    A'B'
        DATA    0
        DATA    A'C'
        DATA    1
        DATA    A'D'
        DATA    2
        DATA    A'E'
        DATA    3
        DATA    A'H'
        DATA    4
        DATA    A'L'
        DATA    5
        DATA    A'M'
        DATA    6
        DATA    0               END OF TABLE INDICATOR
```

THIS ROUTINE SCANS THE INPUT LINE AND PLACES THE OPCODES AND LABELS IN THE BUFFER. THE SCAN TERMINATES WHEN A CHARACTER OTHER THAN 0-9 OR A-Z IS FOUND

```
ALPS    MVI     B,0             SET COUNT
ALP1    STAX    D               STORE CHARACTER IN BUFFER
        INR     B               INCREMENT COUNT
        MOV     A,B             FETCH COUNT
        CPI     11              MAXIMUM BUFFER SIZE
        RNC                     RETURN IF BUFFER FILLED
        INX     D               INCREMENT BUFFER
        INX     H               INCREMENT INPUT ADDRESS
        SHLD    PNTR            SAVE LINE POINTER
        MOV     A,M             FETCH CHARACTER
        CPI     A'0'            CHECK FOR LEGAL CHARACTERS
        RC
        CPI     A'9'+1
        JC      ALP1
        CPI     A'A'
        RC
        CPI     A'Z'+1
        JC      ALP1
        RET
```

```
   THIS ROUTINE IS USED TO CAN THROUGH THE INPUT LINE TO
   FETCH THE VALUE OF THE OPERAND FIELD.  ON RETURN THE VALUE OF THE
   OPERAND IS CONTAINED IN REGISTERS H,L.

SBL     CALL    SBLK            GET FIRST ARGUMENT
 SCN    LXI     H,0             GET A ZERO
        SHLD    OPRD            INITIALIZE OPERAND
        INR     H
        SHLD    OPR-1           INITIALIZE OPERAND INDICATOR
-XT1    LHLD    PNTR            FETCH SCAN POINTER
        DCX     H
        CALL    ZBUF            CLEAR BUFFER
        STA     SIGN            ZERO SIGN INDICATOR
NXT2    INX     H               INCREMENT POINTER
        MOV     A,M             FETCH NEXT CHARACTER
        CPI     A' '+1
        JC      SEND            JUMP IF CR OR BLANK
        CPI     A','            FIELD SEPARATOR
        JZ      SEND
   CHECK FOR OPERATORS
        CPI     A'+'            CHECK FOR PLUS
        JZ      ASC1
        CPI     A'-'            CHECK FOR MINUS
        JNZ     ASC2
        STA     SIGN
ASC1    LDA     OPRI            FETCH OPERAND INDICATOR
        CPI     2               CHECK FOR TWO OPERATORS
        JZ      ERRS            SYNTAX ERROR
        MVI     A,2
        STA     OPRI            SET INDICATOR
        JMP     NXT2
   CHECK FOR OPERANDS
ASC2    MOV     C,A             SAVE CHARACTER
        LDA     OPRI            GET INDICATOR
        ORA     A               CHECK FOR TWO OPERANDS
        JZ      ERRS            SYNTAX ERROR
        MOV     A,C
        CPI     A'$'            LC EXPRESSION
        JNZ     ASC3
        INX     H               INCREMENT POINTER
        SHLD    PNTR            SAVE POINTER
        LHLD    ASPC            FETCH LOCATION COUNTER
        JMP     AVAL
   CHECK FOR ASCII CHARACTERS
ASC3    CPI     A'"'            CHECK FOR SINGLE QUOTE
        JNZ     ASC5            JUMP IF NOT QUOTE
        LXI     D,0             GET A ZERO
        MVI     C,3             CHARACTER COUNT
ASC4    INX     H               INCREMENT POINTER
        SHLD    PNTR            SAVE
        MOV     A,M             FETCH NEXT CHARACTER
        CPI     ASCR            IS IT A CR
        JZ      ERRA            ARGUMENT ERROR
        CPI     A'"'            IS IT A QUOTE
        JNZ     SSTR
```



```
        INX     H               INCREMENT POINTER
        SHLD    PNTR            SAVE
        MOV     A,M             FETCH NEXT CHARACTER
        CPI     A'"'            CHECK FOR 2 QUOTES IN A ROW
        JNZ     AVAL+1          TERMINAL QUOTE
SSTR    DCR     C               CHECK COUNT
        JZ      ERRA            TOO MANY CHARACTERS
        MOV     D,E
        MOV     E,A             SET CHARACTER IN BUFFER
        JMP     ASC4
ASC5    CPI     A'0'            CHECK FOR NUMERIC
        JC      ERRA            ILLEGAL CHARACTER
        CPI     A'9'+1
        JNC     ALAB
        CALL    NUMS            GET NUMERIC VALUE
        JC      ERRA            ARGUMENT ERROR
AVAL    XCHG
        LHLD    OPRD            FETCH OPERAND
        XRA     A               GET A ZERO
        STA     OPRI            STORE IN OPERAND INDICATOR
        LDA     SIGN            GET SIGN INDICATOR
        ORA     A               SET FLAGS
        JNZ     ASUB
        DAD     D               FORM RESULT
ASC7    SHLD    OPRD            SAVE RESULT
        JMP     NXT1
ASUB    MOV     A,L
        SUB     E
        MOV     L,A
        MOV     A,H
        SBB     D
        MOV     H,A
        JMP     ASC7
ALAB    CALL    SLAB
        JZ      AVAL
        JC      ERRA            ILLEGAL SYMBOL
        JMP     ERRU            UNDEFINED SYMBOL

   GET HERE WHEN TERMINATION CHARACTER IS FOUND.  BLANK, COMMA,CR.
   CHECK FOR LEADING FIELD SEPARATOR
SEND    LDA     OPRI            FETCH OPERAND INDICATOR
        ORA     A               SET FLAGS
        JNZ     ERRS            SYNTAX ERROR
        LHLD    OPRD
SEN1    MOV     A,H             GET HIGH ORDER BYTE
        LXI     D,TEMP          SET ADDRESS
        ORA     A               SET FLAGS
        RET

   GET A NUMERIC VALUE WHICH IS EITHER HEXADECIMAL OR DECIMAL
   ON RETURN CARRY SET INDICATES AN ERROR

NUMS    CALL    ALPS            GET NUMERIC
        DCX     D
```



```
        LDAX    D               GET LAST CHARACTER
        LXI     B,ABUF          SET BUFFER ADDRESS
        CPI     A'H'            IS IT HEXADECIMAL
        JZ      NUM2
        CPI     A'D'            IS IT DECIMAL
        JNZ     NUM1
        XRA     A               GET A ZERO
        STAX    D               CLEAR D FROM BUFFER
NUM1    CALL    ADEC            CONVERT DECIMAL VALUE
        RET
NUM2    XRA     A               GET A ZERO
        STAX    D               CLEAR H FROM BUFFER
        CALL    AHEX            CONVERT HEX
        RET

   PROCESS REGISTER ERROR
ERRR    MVI     A,A'R'          GET INDICATOR
        LXI     H,0             GET A 0
        STA     OBUF+18         SET IN OUTPUT BUFFER
        RET
   PROCESS SYNTAX ERROR
ERRS    MVI     A,A'S'          GET INDICATOR
        STA     OBUF+18         STORE IN OUTPUT BUFFER
        LXI     H,0             GET A ZERO
        JMP     SEN1
   PROCESS UNDEFINED SYMBOL ERROR
ERRU    MVI     A,A'U'          GET INDICATOR
        JMP     ERRS+2
   PROCESS VALUE ERROR
ERRV    MVI     A,A'V'          GET INDICATOR
        JMP     ERRR+2
   PROCESS MISSING LABEL ERROR
ERRM    MVI     A,A'M'          GET INDICATOR
        STA     OBUF+18         STORE IN OUTPUT BUFFER
        CALL    AOU1            DISPLAY ERROR
        RET
   PROCESS ARGUMENT ERROR
ERRA    MVI     A,A'A'          GET INDICATOR
        JMP     ERRS+2
   PROCESS OPCODE ERROR
   )RE 3 BYTES OF ZERO IN OBJECT CODE TO PROVIDE FOR A PATCH
   )O   MVI     A,A'O'          GET INDICATOR
        STA     OBUF+18         STORE IN OUTPUT BUFFER
        LDA     PASI            FETCH PASS INDICATOR
        ORA     A               WHICH PASS
        RZ                      RETURN IF PASS 1
        MVI     C,3             NEED 3 BYTES
ERO1    XRA     A                 GET A ZERO
        CALL    ASTO            PUT IN LISTING AND MEMORY
        DCR     C
        JNZ     ERO1
        RET
   PROCESS LABEL ERROR
ERRL    MVI     A,A'L'          GET INDICATOR
```



```
        JMP     ERRO+2
   PROCESS DUPLICATE LABEL ERROR
ERRD    MVI     A,A'D'          GET ERROR INDICATOR
        STA     OBUF+18         STORE IN OUTPUT BUFFER
        CALL    AOU1            DISPLAY ERROR
        JMP     OPC             PROCESS OPCODE
PATCH 01 INX    H
        JNZ     MOV23
        JMP     P+3

   DEFINE INPUT AND OUTPUT PORTS

USTA    EQU     0               UART STATUS
UDAI    EQU     1               DATA IN
UDAO    EQU     1               DATA OUT
PDAI    EQU     6               PROM DATA IN
PADO    EQU     7               PROM ADDRESS OUT
PDAO    EQU     8               PROM DATA OUT
PCTO    EQU     9               PROM CONTROL OUT
SWCH    EQU     H'FF'

   FILE AREA PARAMETERS
MAXFIL  EQU     6               MAX # OF FILES
NMLEN   EQU     5               NAME LENGTH
FELEN   EQU     NMLEN+8         DIRECTORY ENTRY LENGTH
FILE0   RES     NMLEN
BOFP    RES     2
EOFP    RES     2
MAXL    RES     4
FILTB   RES     (MAXFIL-1)*FELEN
INSP    RES     2               INSERT LINE POSITION
DELP    EQU     INSP            DELETE LINE POSITION
ASCR    EQU     13              ASCII CARRIAGE RETURN VALUE
HCON    RES     2
ADDS    EQU     HCON            FIND ADDRESS
FBUF    RES     NMLEN           FILE NAME BUFFER
FREAD   RES     2               FREE ADDRESS IN DIRECTORY
FEF     RES     1               FREE ENTRY FOUND FLAG
FOCNT   EQU     FEF             OUTPUT COUNTER
ABUF    RES     12              ASCII BUFFER
BBUF    RES     4               BINARY BUFFER
SCNT    RES     1

DCNT    RES     1               DUMP ROUTINE COUNTER
NCOM    EQU     10              NUMBER OF COMMANDS
TABA    RES     2               SYMBOL TABLE END ADDRESS
ASPC    RES     2               ASSEMBLER PROGRAM COUNTE
PASI    RES     1               PASS INDICATOR
NCHR    RES     1               LENGTH OF STRING FOR COMPARE
PNTR    RES     2               LINE POINTER STORAGE
NOLA    RES     1               NUMBER OF LABELS
SIGN    RES     1               SIGN STORAGE FOR SCAN
OPRD    RES     2               OPERAND STORAGE
OPRI    RES     1               OPERAND FOUND INDICATOR
```

D000 = FILE0
D005 = EOFP
D007 = MAXL

```
TEMP    RES     1
APNT    EQU     INSP            ASSEMBLE LINE POINTER
AERR    EQU     SCNT            ASSEMBLER ERROR PRINT SWITCH
?IND    RES     2               OUTPUT ADDRESS
.LAB    EQU     5               LENGTH OF LABELS
AREA    RES     18
?BUF    RES     25              OUTPUT BUFFER AREA
        RES     5
IBUF    RES     83
SYMT    EQU     $               START OF SYMBOL TABLE
        END

TOTAL ASSEMBLER ERRORS =   0
```

SYMBOL TABLE

1

A	0007	ABUF	D05A	ACM1	F20E	ACMK	F1FC
ACO1	F755	ACO2	F7D5	ADD5	D050	ADE1	F19D
ADEC	F19A	ADR	F52F	AERR	D06A	AHE1	F1B7
AHEX	F1B4	AHS1	F1CB	ALAB	FC2A	ALP1	FB5A
ALPS	FB5B	AOU1	F6CD	AOU2	F6C7	AOUT	F6C0
APNT	D04E	AREA	D07C	ASBL	FB77	ASC1	FBA7
ASC2	FBB7	ASC3	FBCF	ASC4	FBD9	ASC5	FBFB
ASC7	FC1B	ASCN	FB7A	ASCR	000D	ASM1	F687
ASM2	F6B4	ASM3	F678	ASM4	F668	ASM5	F671
ASPC	D06E	ASSM	F658	ASTO	F8D4	ASUB	FC21
AVAL	FC0B	B	0000	BBUF	D066	BID1	F24D
BIN1	F232	BIND	F23C	BINM	F21F	BLK1	F1F6
BOFP	D005	BSPA	F064	C	0001	CHAR	F06C
CHOT	F1D9	CLER	F086	CO1	F55B	CO2	F561
COM0	F556	COM1	F569	COMM	F0C3	COMS	F0D4
COND	F905	COP1	FA4C	COPC	FA3A	CR	F03C
CRLF	F0A6	CTAB	F257	D	0002	DAT1	F78D
DAT2	F7EB	DCNT	D06B	DEL	F057	DEL1	F5F5
DEL2	F612	DEL3	F630	DEL4	F63A	DEL5	F63B
DELL	F5E1	DELP	D04E	DLA1	F5D7	DLA2	F5D8
DLAY	F5D5	DONE	F158	DOUT	F1EA	DUM1	F285
DUM2	F2C7	DUMP	F29B	DUMS	F2A3	E	0003
EASS	F8EB	EEND	002F	EMES	F426	EMES1	F42C
EMES2	F431	ENT1	F450	ENTR	F437	ENTS	F444
EO1	F528	EOF	F527	EOFP	D007	EOR	F01F
EQU1	F770	EQUL	F4BC	EQUS	F77B	ERO1	FC9C
ERRA	FC8B	ERRD	FCAA	ERRL	FCA5	ERRM	FC82
ERRO	FC90	ERRR	FC64	ERRS	FC6D	ERRU	FC78
ERRV	FC7D	ETRA	F112	EXEC	F0B9	FAST	F3A6
FBUF	D052	FEET	F3B8	FEF	D059	FELEN	000D
FI1	F50E	FI2	F520	FIL3B	F370	FIL3S	F372
FILE	F2FF	FILE0	D000	FILTB	D00D	FIN1	F50B
FIND	F505	FINE	F389	FOCNT	D059	FOOD	F3A3
FOOL	F3C4	FOOT	F383	FOUL	F385	FOUT	F379
FREAD	D057	FSE10	F3E1	FSE15	F404	FSE20	F40E
FSEA	F3D8	GT	F4E4	H	0004	HCON	D050
HOTB	F1E3	HOUT	F1D3	IBUF	D0AC	INB	F08E
INCA	F0F7	INIT2	F006	INITA	F000	INSP	D04E
INSR	F4B3	L	0005	LCHK	F730	LF	F0AB
LICK	F47B	LINE	F476	LIS1	F588	LIST	F584
LLAB	0005	LMOV	F534	LODM	F546	LT	F4D1
M	0006	MAXFIL	0006	MAXL	D009	MESS	F420
MLAB	F700	MOV23	F343	MPNT	F8A5	NCHR	D071
NCOM	000A	NEXT	F02D	NMLEN	0005	NOLA	D074
NOR1	F577	NORM	F571	NOV1	F649	NOVR	F643
NUM1	FC5A	NUM2	FC5E	NUMS	FC46	NXT1	FB84
NXT2	FB8E	OBUF	D08E	OCN1	FADF	OCN2	FAE2
OCNT	FAC8	OERR	FAF0	OIND	D07A	OK	F0A2
OP1	FA6B	OP2	FA6E	OP4	FAC6	OP5	FAC9

OPAD	FAC5	OPC	F719	OPC2	FA73	OPC3	FA8F
OPCD	FA4F	OPRD	D076	OPR1	D078	ORG1	F75B
ORG2	F7F4	OTAB	F911	OUT8	F098	PABL	F7A9
PADO	0007	PAG1	F2EB	PAG2	F2F7	PAGE	F2D8
PAS1	F6D7	PAS2	F790	PAS1	D070	PCTO	0009
PDA1	0006	PDAO	0008	PNTR	D072	PRO1	F5A0
PRO2	F5AE	PRO3	F5CE	PROM	F59D	PSEU	FAF8
PSU1	F73F	PSU2	F7C4	PSW	0006	READ	F025
RES1	F785	RES2	F7DB	RES21	F7E7	RMOV	F53D
ROOM	F339	RTAB	FB47	SBL1	F8FA	SBL2	F8FE
SBLK	F8F7	SCNT	D06A	SCRN	F213	SEAR	F0EB
SEN1	FC40	SEND	FC36	SEOF	F4AC	SIGN	D075
SLA1	FB2E	SLA2	FB41	SLA3	FB44	SLAB	FB0A
SP	0006	SSTR	FBF2	STOM	F54E	SWAP	F350
SWAPS	F34B	SWCH	00FF	SYMT	D0FF	SYS8	F00C
TABA	D06C	TEMP	D079	TEST	F324	TY31	FB3B
TY32	F83E	TY41	F867	TY56	F893	TY6	FBC4
TYP1	F813	TYP2	F817	TYP3	F82A	TYP4	F849
TYP5	F87E	TYP6	F8B4	TYS5	F886	TYS6	FBC7
UDA1	0001	UDAO	0001	USTA	0000	VAL1	F121
VAL2	F13C	VAL3	F14D	VAL4	F14F	VAL5	F15D
VALC	F10B	VCHK	F293	WHA1	F41D	WHAT	F41A
ZBU1	F104	ZBUF	F0FE	ZERO	F4F4	P+3	F34A
PATCH01	FCB6						

ERRATA

SELF-CONTAINED SYSTEM

March 4, 1976

Several bugs have been found in the self-contained system. The fixes are given below:

ADDRESS	IS	SHOULD BE	COMMENTS
04E4	5C	2F	Causes a 'WHAT' response to the ENTR command.
0848 0849	28 0B	A8 08	Causes the assembler to enter labels into the symbol table incorrectly if the DB psuedo-op is used.
0890	4D	81	Causes an 'S' error if DW psuedo-op is used.
091E	FA	C3	Causes an error on INR and DCR. Has been fixed in later releases.
0A32	DF	E9	Causes wrong value to be assembled for PCHL.
0422	84	86	Supposes listings of deleted files.

There are several ambiguities in the documentation.

1. For the ENTR command, data may be entered continuously, on several lines if necessary, until a slash (/) followed by a carriage return is typed. (Note that the fix at 04E4 given above must be made.)

   EXAMPLE:

   ```
   ENTR 500
   0 0A 30 44 FF FE
   89 1E 8E /
   ```

2. In the FILE command, the blank indicated by the arrow below is critical.

   ```
   FILE /name/addr
        ↑
   ```

   If it is left out, nothing will work. Symptons include garbage printout from the ASSM or LIST commands.

IMSAI 8080
Self-Contained System
Assembler
Revision 1

The IMSAI 8080 Monitor, Assembler, and Text Editor, supplied by IMS Associates, Incorporated free of charge, is a modified version of software written by Microtec of Sunnyvale, California for Processor Technology of Berkeley, California who distributed the package free of charge.

IMSAI 8080 SELF-CONTAINED SYSTEM

OPERATING SYSTEM

The IMSAI 8080 Self-Contained System is a software system designed to run on the IMSAI 8080 computer. Included in the package is an Executive to handle memory files, an Assembler, and a line oriented Editor.

To use the system 6K of memory must be available for use by the system. This memory is allocated as follows:

0050 - OEFF	Operating Program
1000 - 1123	Special System RAM
1000 - 17FF	Symbol Table (Assembler Only)

In addition other memory must be available for source and object files necessary for the user's program.

I/O within the program interacts with I/O ports addressed as follows:

PORT	FUNCTION
3	Status Input Bit 6 indicates DAV Bit 7 indicates TBE
2	TTY Input
FF	Sense Switch Input ADDRESS - PROGRAMMED INPUT switch seven is used to control file listing.

Executive Commands

CONTROL X	System reset and CR/LF
ENTR	Enter data to memory
DUMP	Display memory data
FILE	Create, assign or display file information
EXEC	Execute a program
ASSM	Assemble a source file to object code
LIST	List file
DELT	Delete lines of file.
1111	Any four numeric digits enters editor
PAGE	Move a page of data
BREK	Set or clear break points
PROC	Proceed from break point
CUST	Optional user command at location 2000

To initialize the system, start it at 0050. To restart the system without initializing it, start at 0053.

The executive has one error message.....WHAT?.....indicating an improper command or an error on parameters following the command.

Command Format

ENTR AAAA----Enter data to memory

This command is used to enter data to memory starting at address AAAA and continuing until a return command (/) is given. Data is entered in hexadecimal format.

Example:

```
ENTR 500
0 0A 30 44 FF FE/
```

DUMP AAAA BBBB----Dump Contents of Memory

This command is used to examine the contents of memory. The values contained in memory from locations AAAA to BBBB are displayed in hexadecimal. Each line of display consists of an address followed by the contents of the next 16 memory locations. If BBBB is not specified, only location AAAA will be displayed.

FILE /NAME/ AAAA

This command is used to enter, examine or modify parameters of files created in the system. Up to six files can exist simultaneously with any one of the files called as "current". Depending on the form of the command, the following parameters the following functions are performed.

FILE /NAME/ ADDR.	Create a file with the name, NAME starting at address ADDR and make it current. If a file with the same name already exists, output error message NO NO.
FILE /NAME/ 0	Delete file with name NAME and make no file current. Note: No file can start at ADDR 0.
FILE /NAME/	Get file NAME and make it current. Save all parameters of existing current file.
FILE	Display parameters of the "current" file in the following format with AAAA and BBBB being the beginning of file and end of file addresses: NAME AAAA BBBB
FILES	Display the parameters of all files currently saved by the system.

EXEC AAAA----Execute a program

This command is used to execute a program at address AAAA.

LIST N----List file

This command is used to display the lines entered by the user into the file. The output consists of the lines in the file starting at line number N. If N is not specified, the display starts at the beginning of the file. The user can terminate the display by using ADDRESS-PROGRAMMED INPUT switch 7.

DELT L1 L2----Delete line(s) from file

This command is used to delete lines entered by the user from the file. All lines starting at line L1 and continuing up to and including L2 are deleted from the file. If L2 is not specified, only L1 is deleted.

PAGE AAAA BBBB----Move page of data

This command is used to move one page (256 bytes) of data from address AAAA to BBBB.

CUST----Optional user command at location 2000

This command allows any routine to be placed at location 2000 by the user. If the command is terminated by a RET and proper stack operations are used, the system will return in an orderly manner.

BREK or BREK AAAA

This command is used to set or clear break points. If called without the argument AAAA, all break points are cleared.

If called with the argument AAAA, a break point is set at location AAAA. When the break point is encountered in the course of execution, the break point is cleared, all registers are saved, the A register is displayed in the PROGRAMMED OUTPUT on the front panel, the message "AAAA BREAK" is typed and control returns to the executive. The registers are saved in the following locations, and may be examined or modified using the DUMP or ENTR commands.

Location	Register
1000	PSW
1001	A
1002	C
1003	B
1004	E
1005	D
1006	SP (low)
1007	SP (high)
1008	L
1009	H
100A	PC (low)
100B	PC (high)

Restrictions: (1) A maximum of 8 break points may be set.

(2) Break points may not be set below location 000B.

(3) Setting a break point causes information to be stored into locations 0008-000A, destroying any information already there.

PROC or PROC AAAA

This command is used to proceed from a break point. All registers are restored from the locations specified above, and execution continues from the location specified by the PC, unless the argument AAAA is given, in which case execution begins at location AAAA.

ASSM (E) AAAA BBBB ---- Assemble a source file to object code.

This command is used to assemble a source program written by the user and located in the file area. The assembler performs the assembly, assigning addresses to the object code starting at AAAA. On the second pass the object code is placed in memory starting at location BBBB. If BBBB is not specified, it assumes the same value as AAAA. During pass one certain errors are displayed, and during pass two a complete listing is produced. If the optional E is specified in the command, only those lines which contain errors are listed.

TEXT EDITOR

Editor

The editor is a line oriented editor which enables the user to easily create program files in the system. Each line is prefaced by a fixed line number which provides for stable line referencing. Since line numbers can range from 0000 to 9999 (decimal) there are 10,000 lines that can exist in each file. (If enough storage exists.) As the user types lines on the input device, they are entered into the file area. The editor places all line numbers in sequence and automatically over-writes an existing line in the file, if a new line with the same line number is entered by the user. A feature of the editor is that the file area never contains any wasted space.

Note: The Editor ALWAYS operates on the current file.

The editor does not automatically assign line numbers. The user must first, when entering a line of data, enter a decimal number which will be interpreted as being the line number. Valid line numbers must contain four digits... preceding zeros must be included. An entry to the editor is terminated by the carriage return key. No more than 80 characters may be input for one line.

All lines are ordered by the ascending numeric sequence of their line numbers. If the user wishes to insert lines after the initial entry is made, it is suggested that he input the original lines with line numbers at least five units difference.

ASSEMBLER

When the Assembler is given control by the executive, it proceeds to translate the Symbolic 8080 Assembly Language (Source) program into 8080 machine (object) code. The Assembler is a two pass assembler which operates on the "current" file. Features of the Assembler include:

- free format source input.
- symbolic addressing, including forward references and relative symbolic references.
- complex expressions may be used as arguments.
- self defining constants.
- multiple constant forms.
- up to 256 five character symbols.
- reserved names for 8080 registers
- ASCII character code generation
- 6 Pseudo Operations (assembler directives)

The assembler translates those lines contained in the current file into object code. The second character following the line number, is considered to be the first source code character position. Hence, the character immediately following the line number should normally be blank. Line numbers are not processed by the assembler; they are merely reproduced on the listing.

The assembler will assemble a source program file composed of STATEMENTS, COMMENTS, and PSEUDO OPERATIONS.

During Pass 1, the assembler allocates all storage necessary for the translated program and defines the values of all symbols used, by creating a symbol table. The storage allocated for the object code will begin at the first byte dictated by the 1st parameter in the original Executive ASSM command.

During Pass 2, all expressions, symbols and ASCII constants are evaluated to absolute values and are placed in allocated memory in the appropriate locations. The listing, also produced during Pass 2, indicates exactly what data is in each location of memory.

Statements

Statements may contain either symbolic 8080 machine instructions or pseudo-ops. The structure of such a statement is:

```
NAME     OPERATION     OPERAND     COMMENT
```

The name-field, if present, must begin in assembler character position one. The symbol in the name field can contain as many characters as the user wants; however, only the first 5 characters are used in the symbol table to uniquely define a symbol. All symbols in this field must begin with an alphabetic character and may contain no special characters.

The operation field contains either a 8080 operation mnemonic or a system pseudo-operation code.

The operand field contains parameters pertaining to the operation in the operation field. If two arguments are present, they must be separated by a comma. Example:

```
0015 FLOP  MOV M,B  COMMENT
0020 * COMMENT
0025       JMP  BEG
0030       CALL FLOP
0035 BEG   ADI  8+6-4
0040       MOV  A,B
```

All fields are separated and distinguished from one another by the presence of one or more blank characters (spaces).

The comment field is for explanatory remarks. It is reproduced on the listing without processing. See example 0015. Comment lines must start with an asterisk (*) in character position 1. See example 0020.

Symbolic Names

To assign a symbolic name to a statement, one merely places the symbol in the name field. To leave off the name field, the user skips two or more spaces after the line number and begins the operation field. If a name is attached to a statement, the assembler assigns it the value of the current Location Counter. The Location Counter always holds the address of the next byte to be assembled. The only exception to this is the EQU pseudo-op. In this case

a symbol in the name field is assigned a value which is contained in the operand field of the EQU pseudo-op statement. Example:

```
0057 POTTS EQU 128
```

assigns the value 128 to the name POTTS. This data can then be used elsewhere in the program as: eg ADI POTTS.

Names are defined when they appear in the name field. All defined names may be used as symbolic arguments in the argument field. See examples 0015, 0025, 0030, 0035.

In addition to user defined names, the assembler has reserved several symbols, the value of which is predetermined. These names may not be used by the user except in the operand field. They are (with their value in parenthesis):

```
A- the accumulator    (7)
B- Register B         (0)
C- Register C         (1)
D- Register D         (2)
E- Register E         (3)
H- Register H         (4)
L- Register L         (5)
M- Memory (through H,L)  (6)
```

In addition to the above reserved symbols, there is the single special character symbol ($). This symbol changes in value as the assembly progresses. It is alway equated with the value of the program counter after the current instruction is assembled. It may only be used in the operand field. Examples:

```
JMP  $        means jump to the next location.
MOV  A,B      after this instruction; i.e., the
              MOV instruction.
LDA  $+5
DB    0
DB    1       means load the data at the fifth location
DB    2       after this location.  In this case, the
DB    3       data has the value 5.
DB    4
DB    5
```

Relative Symbolic Addressing

If the name of a particular location is known, a nearby location may be specified using the known name and a numeric offset. Example:

```
      JMP   BEG
      JPE   BEG+4
      CC    SUB
      CALL  $+48
BEG   MOV   A,B
      HALT
      MVI   C, 'B'
      INR   B
```

In this example the instruction JMP BEG refers to the MOV A,B instruction. The instruction JPE BEG+4 refers to the INR B instruction. BEG+4 means the address BEG plus four bytes. This form of addressing can be used to locate several bytes before or after a named location.

Constants

The Assembler allows the user to write positive or negative numbers directly in a statement. They will be regarded as decimal constants and their binary equivalents will be used appropriately. All unsigned numbers are considered positive. Decimal constants can be defined using the descriptor "D" after the numeric value. (This is not required, as the default is decimal.)

Hexadecimal constants may be defined using the descriptor "H" after a numeric value. IE. +10H, 10H, 3AH, 0F4H.

Note that a hexadecimal constant cannot start with the digits A-F. In this case, a leading 0 must be included. This enables the assembler to differentiate between a numeric value and a symbol.

ASCII constants may be defined by enclosing the ASCII character within single quote marks, i.e., 'C'. For double word constants, two characters may be defined within one quote string.

Expressions

An expression is a sequence of one or more symbols, constants or other expressions separated by the arithmetic operators plus or minus.

```
PAM +3
ISAB-'A'+52
LOOP+32H-5
```

Expressions are calculated using 16 bit arithmetic. All arithmetic is done modulo 65536. Single byte data cannot contain a value greater than 255 or less than -256. Any value outside this range will result in an assembler error.

Pseudo-Operations

The pseudo-operations are written as ordinary statements, but they direct the assembler to perform certain functions which do not always develop 8080 machine code. The following section describes the pseudo-ops.

ORG----Set Program Origin

Format is

```
label ORG expression
```

where the label is optional but if present will be equaled to the given expression.

END----End of Assembly

The pseudo-op informs the assembler that the last source statement has been read. The assembler will then start on pass 2 and terminate the assembly and pass control back to the executive. This pseudo-op is not needed when assembling from a memory file since the assembler will stop when an end of file indicator has been reached.

EQU----Equal Symbolic Value

Format is

label EQU expression

where label is a symbol the value of which will be determined from the expression, and expression is an expression which when evaluated will be assigned to the symbol given in the name field.

DS----Define Storage

Format is

label DS expression.

The DS causes the assembler to advance the Assembly Program Counter, effectively skipping past a given number of memory bytes.

DB----Define Byte

Format is

label DB expression.

This pseudo-op is used to reserve one byte of storage. The content of the byte is specified in the argument field.

DW----Define Word

This pseudo-op is used to define two bytes of storage. The evaluated argument will be placed in the two bytes; high order 8 bits in the low order byte, and the low order 8 bits in the high order byte. This conforms to the Intel format for two byte addresses.

Assembler Errors

The following error flags are output on the assembler listing when the error occurs. Some of the errors are only output during pass 1.

0	Opcode Error
L	Label Error
D	Duplicate Label Error
M	Missing Label Error
V	Value Error
U	Undefined Symbol
S	Snytax Error
R	Register Error
A	Argument Error.

OBJECT TAPE FORMAT

The IMSAI Self-Contained System is supplied on paper tape in a blocked hexadecimal format. The data on the tape is blocked into discrete records, each record containing record length, record type, memory address and checksum information in addition to data. A frame-by-frame description is as follows:

Frame 0	Record Mark. Signals the start of a record. The ASCII character colon (":" HEX 3A) is used as the record mark.
Frames 1,2 (0-9,A-F)	Record Length. Two ASCII characters representing a hexadecimal number in the range 0 to 'FF' (0 to 255). This is the count of actual data bytes in the record type or checksum. A record length of 0 indicates end of file.
Frames 3 to 6	Load Address. Four ASCII characters that represent the initial memory location where the data following will be loaded. The first data byte is stored in the location pointed to by the load address; succeeding data bytes are loaded into ascending addresses.
Frames 7, 8	Record Type. Two ASCII characters. Currently all records are type 0. This field is reserved for future expansion.
Frames 9 to 9+2* (Record Length) -1	Data. Each 8 bit memory word is represented by two frames containing the ASCII characters (0 to 9, A to F) to represent a hexadecimal value 0 to 'FF'H (0 to 255).

Frames 9+2* (Record Length) to 9+2*(Record Length) +1	Checksum. The checksum is the negative of the sum of all 8 bit bytes in the record since the record mark (":") evaluated modulus 256. That is, if you add together all the 8 bit bytes, ignoring all carries out of an 8-bit sum, then add the checksum, the result is zero.

Example: If memory locations 1 through 3 contain 53F8EC, the format of the hex file produced when these locations are punched is:

```
:0300010053F8ECC5
```

SAVING AND RESTORING PROGRAMS

While the system has no explicit provision for saving and restoring programs, it is possible to do so with an ASR style teletype. The procedure is as follows:

1. Make the file you want to save the current file.
2. Type 'LIST', but don't type the carriage return.
3. Turn on the paper tape punch.
4. Type carriage return. The program will be listed on the teletype and simultaneously punched on the paper tape punch.
5. When the 'LIST' is completed, turn off the punch.

The procedure for restoring the file is as follows:

1. Make the file you want to restore into the current file.
2. Mount the tape in the paper tape reader.
3. Start the paper tape reader. The program will be automatically read in.

```
                        ; REVISION 2                06 OCT 76
                        ;
                        ; ********* SELF CONTAINED SYSTEM *********
                        ;
0000                            ORG     00H
0000 C34000                     JMP     INITA   ;DEAD START
0003 C36700                     JMP     EOR     ;RESTART MONITOR
                        ;
0006                            ORG     08H
0008 C32E0D                     JMP     BRKP    ;BREAKPOINT RESTART
                        ;
0008                            ORG     40H
                        ;
                        ; THIS ROUTINE SETS UP THE SIO BOARD
                        ;
0040 3EAA               INITA:  MVI     A,0AAH  ;GET DUMMY MODE WORD
0042 D303                       OUT     TTS     ;OUTPUT IT
0044 3E40                       MVI     A,40H   ;GET RESET BIT
0046 D303                       OUT     TTS     ;RESET SIO BOARD
0048 3ECE                       MVI     A,0CEH  ;GET REAL MODE WORD
004A D303                       OUT     TTS     ;SET THE MODE FOR REAL
004C 3E37                       MVI     A,37H   ;GET THE COMMAND
004E D303                       OUT     TTS     ;OUTPUT IT
                        ;
                        ; THIS ROUTINE INITIALIZES THE FILE AREA FOR SUBSEQUENT
                        ; PROCESSING
                        ;
0050 212410                     LXI     H,FILE0
0053 0E4E                       MVI     C,MAXFIL*FELEN
0055 AF                         XRA     A
0056 77                 INIT2:  MOV     M,A
0057 23                         INX     H
0058 0D                         DCR     C
0059 C25600                     JNZ     INIT2
                        ;
                        ; CLEAR THE BREAKPOINT TABLE
                        ;
005C 0618                       MVI     B,NBR*3
005E 210C10                     LXI     H,BRT
0061 77                 INIT3:  MOV     M,A
0062 23                         INX     H
0063 05                         DCR     B
0064 C26100                     JNZ     INIT3
                        ;
                        ; THIS IS THE STARTING POINT OF THE SELF CONTAINED
                        ; SYSTEM ONCE THE SYSTEM HAS BEEN INITIALIZED.  COMMANDS
                        ; ARE READ FROM THE USER, EXECUTED, AND CONTROL RETURNS
                        ; BACK TO THIS POINT TO READ ANOTHER COMMAND.
                        ;
0067 31B210             EOR:    LXI     SP,AREA+18
006A CD0E01                     CALL    CRLF    ;PRINT C/R, LINE FEED
006D CD8000                     CALL    READ    ;READ INPUT LINE
0070 23                         INX     H
0071 7E                         MOV     A,M     ;FETCH FIRST CHARACTER
0072 FE3A                       CPI     '9'+1   ;COMMAND OR LINE NUMBER?
0074 DAB504                     JC      LINE    ;JUMP IF LINE FOR FILE
0077 CD7301                     CALL    VALC    ;GET COMMAND VALUES
007A CD2B01                     CALL    COMM    ;CHECK LEGAL COMMANDS
007D C36700                     JMP     EOR
                        ;
                        ; THIS ROUTINE READS IN A LINE FROM THE TTY AND PLACES
                        ; IT IN AN INPUT BUFFER.
                        ; THE FOLLOWING ARE SPECIAL CHARACTERS
                        :   CR          TERMINATES READ ROUTINE
```

```
                        ;
                        ;    LF             NOT RECOGNIZED BY ROUTINE
                        ;    CTRL X         DELETE CURRENT LINE
                        ;    DEL            DELETE CHARACTER
                        ; ALL DISPLAYABLE CHARACTERS BETWEEN BLANK & Z AND THE
                        ; ABOVE ARE RECOGNIZED BY THE READ ROUTINE, ALL OTHERS
                        ; ARE SKIPPED OVER.  THE ROUTINE WILL NOT ACCEPT MORE
                        ; CHARACTERS THAN THE INPUT BUFFER WILL HOLD.
                        ;
0080 21C710     READ:   LXI     H,IBUF  ;GET INPUT BUFFER ADDRESS
0083 227410             SHLD    ADDS    ;SAVE ADDRESS
0086 1E02               MVI     E,2     ;INITIALIZE CHARACTER COUNT
0088 CDF600     NEXT:   CALL    IN8     ;READ A LINE
008B 78                 MOV     A,B
008C FE18               CPI     24      ;CHECK FOR CTRL X
008E C29700             JNZ     CR
0091 CD0E01             CALL    CRLF    ;OUTPUT A CRLF
0094 C38000             JMP     READ
0097 FE0D       CR:     CPI     ASCR    ;GET AN ASCII CR
0099 C2B200             JNZ     DEL
009C 7D                 MOV     A,L
009D FEC7               CPI     IBUF AND 0FFH   ;CHECK FOR FIRST CHAR
009F CA8000             JZ      READ
00A2 360D               MVI     M,ASCR  ;PLACE CR AT END OF LINE
00A4 23                 INX     H
00A5 3601               MVI     M,1     ;PLACE EOF INDICATOR IN LINE
00A7 23                 INX     H
00A8 3E1A               MVI     A,IBUF+83 AND 0FFH
00AA CDE100             CALL    CLER    ;CLEAR REMAINING BUFFER
00AD 21C610             LXI     H,IBUF-1
00B0 73                 MOV     M,E     ;SAVE CHARACTER COUNT
00B1 C9                 RET
00B2 FE7F       DEL:    CPI     127     ;CHECK FOR DELETE CHARACTER
00B4 C2C700             JNZ     CHAR
00B7 3EC7               MVI     A,IBUF AND 0FFH
00B9 BD                 CMP     L       ;IS THIS 1ST CHARACTER
00BA CA8800             JZ      NEXT
00BD 2B                 DCX     H       ;DECREMENT POINTER
00BE 1D                 DCR     E       ;DECREMENT COUNT
00BF 065F       BSPA:   MVI     B,5FH
00C1 CD0301             CALL    OUT8
00C4 C38800             JMP     NEXT
00C7 FE20       CHAR:   CPI     ' '     ;CHECK FOR LEGAL CHARACTER
00C9 DA8800             JC      NEXT
00CC FE5B               CPI     'Z'+1
00CE D28800             JNC     NEXT
00D1 47                 MOV     B,A
00D2 CD0301             CALL    OUT8    ;ECHO CHARACTER
00D5 70                 MOV     M,B
00D6 3E18               MVI     A,IBUF+81 AND 0FFH
00D8 BD                 CMP     L       ;CHECK FOR END OF LINE
00D9 CABF00             JZ      BSPA
00DC 23                 INX     H
00DD 1C                 INR     E       ;INCREMENT CHARACTER COUNT
00DE C38800             JMP     NEXT
                        ;
                        ;
                        ;
                        ; THIS ROUTINE IS USED TO BLANK OUT A PORTION OF MEMORY
                        ;
00E1 BD         CLER:   CMP     L
00E2 C8                 RZ
00E3 3620               MVI     M,' '   ;PLACE BLANK IN MEMORY
00E5 23                 INX     H
00E6 C3E100             JMP     CLER
                        ;
                        ; SEE IF TTY INPUT READY AND CHECK FOR CTRL X.
```

```
                        ;
                        ; RETURN WITH ZERO SET IFF CTRL X SEEN.
                        ;
00E9 DB03       INK:    IN      TTS     ;GET TTY STATUS
00EB 2F                 CMA             ;INVERT STATUS
00EC E602               ANI     TTYDA   ;IS DATA AVAILABLE?
00EE C0                 RNZ             ;RETURN IF NOT
00EF DB02               IN      TTI     ;GET THE CHAR
00F1 E67F               ANI     07FH    ;STRIP OFF PARITY
00F3 FE18               CPI     'X'-40H ;IS IT A CTRL X?
00F5 C9                 RET
                        ;
                        ; THIS ROUTINE READS A BYTE OF DATA FROM THE USART
                        ;
00F6 DB03       IN8:    IN      TTS     ;READ USART STATUS
00F8 E602               ANI     TTYDA
00FA CAF600             JZ      IN8
00FD DB02               IN      TTI     ;READ DATA
00FF E67F               ANI     127     ;STRIP OFF PARITY
0101 47                 MOV     B,A
0102 C9                 RET
                        ;
                        ; THIS ROUTINE OUTPUTS A BYTE OF DATA TO THE USART
                        ;
0103 DB03       OUT8:   IN      TTS     ;READ STATUS
0105 E601               ANI     TTYTR
0107 CA0301             JZ      OUT8
010A 78         OK:     MOV     A,B
010B D302               OUT     TTO     ;TRANSMIT DATA
010D C9                 RET
                        ;
                        ; THIS ROUTINE WILL OUTPUT A CARRIAGE RETURN AND
                        ; LINE FEED FOLLOWED BY TWO DELETE CHARACTERS WHICH
                        ; PROVIDE TIME FOR PRINT HEAD TO RETURN.
                        ;
010E 060D       CRLF:   MVI     B,13    ;CR
0110 CD0301             CALL    OUT8
0113 060A       LF:     MVI     B,10    ;LF
0115 CD0301             CALL    OUT8
0118 067F               MVI     B,127
011A CD0301             CALL    OUT8
011D CD0301             CALL    OUT8
0120 C9                 RET
                        ;
                        ; THIS ROUTINE JUMPS TO A LOCATION IN MEMORY GIVEN BY
                        ; THE INPUT COMMAND AND BEGINS EXECUTION OF PROGRAM
                        ; THERE.
                        ;
0121 CD0003     EXEC:   CALL    VCHK    ;CHECK FOR PARAMETER
0124 CD0E01             CALL    CRLF
0127 2A8A10             LHLD    BBUF    ;FETCH ADDRESS
012A E9                 PCHL            ;JUMP TO PROGRAM
                        ;
                        ;
                        ;
                        ;
                        ; THIS ROUTINE CHECKS THE INPUT COMMAND AGAINST ALL
                        ; LEGAL COMMANDS STORED IN A TABLE.  IF A LEGAL COMMAND
                        ; IS FOUND, A JUMP IS MADE TO THAT ROUTINE.  OTHERWISE
                        ; AN ERROR MESSAGE IS OUTPUT TO THE USER.
                        ;
012B 11BE02     COMM:   LXI     D,CTAB  ;COMMAND TABLE ADDRESS
012E 060B               MVI     B,NCOM  ;NUMBER OF COMMANDS
0130 3E04               MVI     A,4     ;LENGTH OF COMMAND
0132 329510             STA     NCHR    ;SAVE
0135 CD3C01             CALL    COMS    ;SEARCH TABLE
0138 C25A04             JNZ     WHAT    ;JUMP IF ILLEGAL COMMAND
```

```
013B E9                 PCHL            ;BE HERE NOW
                        ;
                        ;
                        ;
                        ;
                        ;
                        ; THIS ROUTINE CHECKS TO SEE IF A BASE CHARACTER STRING
                        ; IS EQUAL TO ANY OF THE STRINGS CONTAINED IN A TABLE
                        ; POINTED TO BY D,E.  THE TABLE CONSISTS OF ANY NUMBER
                        ; OF CHARS, WITH 2 BYTES CONTAINING VALUES ASSOCIATED
                        ; WITH IT.  REG B CONTAINS THE # OF STRINGS TO COMPARE.
                        ; THIS ROUTINE CAN BE USED TO SEARCH THROUGH A COMMAND
                        ; OR SYMBOL TABLE.  ON RETURN, IF THE ZERO FLAG IS SET,
                        ; A MATCH WAS FOUND; IF NOT, NO MATCH WAS FOUND.  IF
                        ; A MATCH WAS FOUND, D,E POINT TO THE LAST BYTE
                        ; ASSOCIATED WITH THE CHARACTER STRING.  IF NOT, D,E
                        ; POINT TO THE NEXT LOCATION AFTER THE END OF THE TABLE.
                        ;
013C 2A7410     COMS:   LHLD    ADDS    ;FETCH COMPARE ADDRESS
013F 3A9510             LDA     NCHR    ;GET LENGTH OF STRING
0142 4F                 MOV     C,A
0143 CD5301             CALL    SEAR    ;COMPARE STRINGS
0146 1A                 LDAX    D       ;FETCH VALUE
0147 6F                 MOV     L,A
0148 13                 INX     D
0149 1A                 LDAX    D       ;FETCH VALUE
014A 67                 MOV     H,A
014B C8                 RZ
014C 13                 INX     D       ;SET TO NEXT STRING
014D 05                 DCR     B       ;DECREMENT COUNT
014E C23C01             JNZ     COMS
0151 04                 INR     B       ;CLEAR ZERO FLAG
0152 C9                 RET
                        ;
                        ;
                        ;
                        ; THIS ROUTINE CHECKS TO SEE IF TWO CHARACTER STRINGS IN
                        ; MEMORY ARE EQUAL.  THE STRINGS ARE POINTED TO BY D,E
                        ; AND H,L.  ON RETURN, THE ZERO FLAG SET INDICATES A
                        ; MATCH.  REG C INDICATES THE LENGTH OF THE STRINGS. ON
                        ; RETURN, THE POINTERS POINT TO THE NEXT ADDRESS AFTER
                        ; THE CHARACTER STRINGS.
                        ;
0153 1A         SEAR:   LDAX    D       ;FETCH CHARACTER
0154 BE                 CMP     M       ;COMPARE CHARACTERS
0155 C25F01             JNZ     INCA
0158 23                 INX     H
0159 13                 INX     D
015A 0D                 DCR     C       ;DECREMENT CHARACTER COUNT
015B C25301             JNZ     SEAR
015E C9                 RET
015F 13         INCA:   INX     D
0160 0D                 DCR     C
0161 C25F01             JNZ     INCA
0164 0C                 INR     C       ;CLEAR ZERO FLAG
0165 C9                 RET
                        ;
                        ; THIS ROUTINE ZEROES OUT A BUFFER IN MEMORY WHICH IS
                        ; THEN USED BY OTHER SCANNING ROUTINES.
                        ;
0166 AF         ZBUF:   XRA     A       ;GET A ZERO
0167 118A10             LXI     D,ABUF+12 ;BUFFER ADDRESS
016A 060C               MVI     B,12    ;BUFFER LENGTH
016C 1B         ZBU1:   DCX     D       ;DECREMENT ADDRESS
016D 12                 STAX    D       ;ZERO BUFFER
016E 05                 DCR     B
```

```
016F C26C01             JNZ     ZBU1
0172 C9                 RET
                ;
                ; THIS ROUTINE CALLS ETRA TO OBTAIN THE INPUT PARAMETER
                ; VALUES AND CALLS AN ERROR ROUTINE IF AN ERROR OCCURRED
                ; IN THAT ROUTINE.
                ;
0173 CD7A01     VALC:   CALL    ETRA    ;GET INPUT PARAMETERS
0176 DA5A04             JC      WHAT    ;JUMP IF ERROR
0179 C9                 RET
                ;
                ; THIS ROUTINE EXTRACTS THE VALUES ASSOCIATED WITH A
                ; COMMAND FROM THE INPUT STREAM AND PLACES THEM IN THE
                ; ASCII BUFFER (ABUF).  IT ALSO CALLS A ROUTINE TO
                ; CONVERT THE ASCII HEXADECIMALS TO BINARY AND STORES
                ; THEM IN THE BINARY BUFFER (BBUF).  ON RETURN, CARRY
                ; SET INDICATES AN ERROR IN INPUT PARAMETERS.
                ;
017A 210000     ETRA:   LXI     H,0     ;GET A ZERO
017D 228C10             SHLD    BBUF+2  ;ZERO VALUE
0180 227610             SHLD    FBUF    ;SET NO FILE NAME
0183 CD6601             CALL    ZBUF    ;ZERO BUFFER
0186 21C610             LXI     H,IBUF-1
0189 23         VAL1:   INX     H
018A 7E                 MOV     A,M     ;FETCH INPUT CHARACTER
018B FE20               CPI     ' '     ;LOOK FOR FIRST CHARACTER
018D 3F                 CMC
018E D0                 RNC             ;RETURN IF NO CARRY
018F C28901             JNZ     VAL1    ;JUMP IF NO BLACK
0192 229610             SHLD    PNTR    ;SAVE POINTER
0195 CD0D09             CALL    SBLK    ;SCAN TO FIRST PARAMETER
0198 3F                 CMC
0199 D0                 RNC             ;RETURN IF CR
019A FE2F               CPI     '/'
019C C2C401             JNZ     VAL5    ;NO FILE NAME
019F 117610             LXI     D,FBUF  ;NAME FOLLOWS PUT IN FBUF
01A2 0E05               MVI     C,NMLEN
01A4 23         VAL2:   INX     H
01A5 7E                 MOV     A,M
01A6 FE2F               CPI     '/'
01A8 CAB401             JZ      VAL3
01AB 0D                 DCR     C
01AC FA5A04             JM      WHAT
01AF 12                 STAX    D       ;STORE FILE NAME
01B0 13                 INX     D
01B1 C3A401             JMP     VAL2
01B4 3E20       VAL3:   MVI     A,' '   ;GET AN ASCII SPACE
01B6 0D         VAL4:   DCR     C
01B7 FABF01             JM      DONE
01BA 12                 STAX    D       ;FILL IN WITH SPACES
01BB 13                 INX     D
01BC C3B601             JMP     VAL4
01BF CD1409     DONE:   CALL    SBL2
01C2 3F                 CMC
01C3 D0                 RNC
01C4 117E10     VAL5:   LXI     D,ABUF
01C7 CD750B             CALL    ALPS    ;PLACE PARAMETER IN BUFFER
01CA 78                 MOV     A,B     ;GET DIGIT COUNT
01CB FE05               CPI     5       ;CHECK NUMBER OF DIGITS
01CD 3F                 CMC
01CE D8                 RC              ;RETURN IF TOO MANY DIGITS
01CF 017E10             LXI     B,ABUF
01D2 CD1B02             CALL    AHEX    ;CONVERT VALUE
01D5 D8                 RC              ;ILLEGAL CHARACTER
01D6 228A10             SHLD    BBUF    ;SAVE   IN BINARY BUFFER
01D9 217E10             LXI     H,ABUF
```

```
01DC CDBD05           CALL    NORM    ;NORMALIZE ASCII VALUE
01DF CD0D09           CALL    SBLK    ;SCAN TO NEXT PARAMETER
01E2 3F               CMC
01E3 D0               RNC             ;RETURN IF CR
01E4 118210           LXI     D,ABUF+4
01E7 CD750B           CALL    ALPS    ;PLACE PARAMETER IN BUFFER
01EA 78               MOV     A,B     ;GET DIGIT COUNT
01EB FE05             CPI     5       ;CHECK NUMBER OF DIGITS
01ED 3F               CMC
01EE D8               RC              ;RETURN IF TOO MANY DIGITS
01EF 018210           LXI     B,ABUF+4
01F2 CD1B02           CALL    AHEX    ;CONVERT VALUE
01F5 D8               RC              ;ILLEGAL VALUE
01F6 228C10           SHLD    BBUF+2  ;SAVE IN BINARY BUFFER
01F9 218210           LXI     H,ABUF+4
01FC CDBD05           CALL    NORM    ;NORMALIZE ASCII VALUE
01FF B7               ORA     A       ;CLEAR CARRY
0200 C9               RET
                ;
                ; THIS ROUTINE FETCHES DIGITS FROM THE BUFFER ADDRESSED
                ; BY B,C AND CONVERTS THE ASCII DECIMAL DIGITS INTO
                ; BINARY.  UP TO A 16-BIT VALUE CAN BE CONVERTED.  THE
                ; SCAN STOPS WHEN A BINARY ZERO IS FOUND IN THE BUFFER.
                ;
0201 210000     ADEC:   LXI     H,0     ;GEP A 16 BIT ZERO
0204 0A         ADE1:   LDAX    B       ;FETCH ASCII DIGIT
0205 B7                 ORA     A       ;SET ZERO FLAG
0206 C8                 RZ              ;RETURN IFF FINISHED
0207 54                 MOV     D,H     ;SAVE CURRENT VALUE
0208 5D                 MOV     E,L     ;SAVE CURRENT VALUE
0209 29                 DAD     H       ;TIMES TWO
020A 29                 DAD     H       ;TIMES TWO
020B 19                 DAD     D       ;ADD IN ORIGINAL VALUE
020C 29                 DAD     H       ;TIMES TWO
020D D630               SUI     48      ;ASCII BIAS
020F FE0A               CPI     10      ;CHECK FOR LEGAL VALUE
0211 3F                 CMC
0212 D8                 RC              ;RETURN IF ERROR
0213 5F                 MOV     E,A
0214 1600               MVI     D,0
0216 19                 DAD     D       ;ADD IN NEXT DIGIT
0217 03                 INX     B       ;INCREMENT POINTER
0218 C30402             JMP     ADE1
                ;
                ;
                ; THIS ROUTINE FETCHES DIGITS FROM THE BUFFER ADDRESSED
                ; BY B,C AND CONVERTS THE ASCII HEXADECIMAL DIGITS INTO
                ; BINARY.  UP TO A 16-BIT VALUE CAN BE CONVERTED.  THE
                ; SCAN STOPS WHEN A BINARY ZERO IS FOUND IN THE BUFFER.
                ;
021B 210000     AHEX:   LXI     H,0     ;GET A 16 BIT ZERO
021E 0A         AHE1:   LDAX    B       ;FETCH ASCII DIGIT
021F B7                 ORA     A       ;SET ZERO FLAG
0220 C8                 RZ              ;RETURN IF DONE
0221 29                 DAD     H       ;LEFT SHIFT
0222 29                 DAD     H       ;LEFT SHIFT
0223 29                 DAD     H       ;LEFT SHIFT
0224 29                 DAD     H       ;LEFT SHIFT
0225 CD3202             CALL    AHS1    ;CONVERT TO BINARY
0228 FE10               CPI     10H     ;CHECK FOR LEGAL VALUE
022A 3F                 CMC
022B D8                 RC              ;RETURN IF ERROR
022C 85                 ADD     L
022D 6F                 MOV     L,A
022E 03                 INX     B       ;INCREMENT POINTER
022F C31E02             JMP     AHE1
```

```
                        ;
                        ; THIS SUBROUTINE CONVERTS ASCII HEX DIGITS INTO BINARY
                        ;
0232 D630       AHS1:   SUI     48      ;ASCII BIAS
0234 FE0A               CPI     10      ;DIGIT 0-10
0236 D8                 RC
0237 D607               SUI     7       ;ALPHA BIAS
0239 C9                 RET
                        ;
                        ;
                        ; THIS ROUTINE CONVERTS A BINARY VALUE TO ASCII
                        ; HEXADECIMAL AND OUTPUTS THE CHARACTERS TO THE TTY.
                        ;
023A CD8602     HOUT:   CALL    BINH    ;CONVERT VALUE
023D 217410             LXI     H,HCON  ;CONVERSION AREA
0240 46         CHOT:   MOV     B,M     ;FETCH OUTPUT CHARACTER
0241 CD0301             CALL    OUT8    ;OUTPUT CHARACTER
0244 23                 INX     H
0245 46                 MOV     B,M     ;FETCH CHARACTER
0246 CD0301             CALL    OUT8    ;OUTPUT CHARACTER
0249 C9                 RET
                        ;
                        ; THIS ROUTINE DOES THE SAME AS ABOVE BUT OUTPUTS A
                        ; BLANK AFTER THE LAST CHARACTER
                        ;
024A CD3A02     HOTB:   CALL    HOUT    ;CONVERT AND OUTPUT
024D CD5D02             CALL    BLK1    ;OUTPUT A BLANK
0250 C9                 RET
                        ;
                        ;
                        ; THIS ROUTINE CONVERTS A BINARY VALUE TO ASCII
                        ; DECIMAL DIGITS AND OUTPUTS THE CHARACTERS TO THE TTY
                        ;
0251 CDA302     DOUT:   CALL    BIND    ;CONVERT VALUE
0254 CD3D02             CALL    HOUT+3  ;OUTPUT VALUE (2 DIGITS)
0257 23                 INX     H
0258 46                 MOV     B,M     ;GET LAST DIGIT
0259 CD0301             CALL    OUT8    ;OUTPUT
025C C9                 RET
                        ;
                        ; THIS ROUTINE OUTPUTS A BLANK
                        ;
025D 0620       BLK1:   MVI     B,' '   ;GET A BLANK
025F CD0301             CALL    OUT8
0262 C9                 RET
                        ;
                        ;
                        ; THIS ROUTINE IS USED BY OTHER ROUTINES TO INCREMENT
                        ; THE STARTING ADDRESS IN A COMMAND AND COMPARE IT WITH
                        ; THE FINAL ADDRESS IN THE COMMAND.  ON RETURN, THE
                        ; CARRY FLAG SET INDICATES THAT THE FINAL ADDRESS HAS
                        ; BEEN REACHED.
                        ;
0263 2A8A10     ACHK:   LHLD    BBUF    ;FETCH START ADDRESS
0266 3A8D10             LDA     BBUF+3  ;STOP ADRESS (HIGH)
0269 BC                 CMP     H       ;COMPARE ADDRESSES
026A C27502             JNZ     ACH1
026D 3A8C10             LDA     BBUF+2  ;STOP ADDRESS (LOW)
0270 BD                 CMP     L       ;COMPARE ADDRESSES
0271 C27502             JNZ     ACH1
0274 37                 STC             ;SET CARRY IF EQUAL
0275 23         ACH1:   INX     H       ;INCREMENT START ADDRESS
0276 228A10             SHLD    BBUF    ;STORE START ADDRESS
0279 C9                 RET
                        ;
                        ;
```

```
                        ;
                        ; THIS ROUTINE OUTPUTS CHARACTERS OF A STRING
                        ; UNTIL A CARRIAGE RETURN IS FOUND.
                        ;
027A 46         SCRN:   MOV     B,M     ;FETCH CHARACTER
027B 3E0D               MVI     A,13    ;CARRIAGE RETURN
027D B8                 CMP     B       ;CHARACTER = CR?
027E C8                 RZ
027F CD0301             CALL    OUT8    ;OUTPUT CHARACTER
0282 23                 INX     H       ;INCREMENT ADDRESS
0283 C37A02             JMP     SCRN
                        ;
                        ;
                        ; THIS ROUTINE CONVERTS THE BINARY VALUE IN REG A INTO
                        ; ASCII HEXADECIMAL DIGITS AND STORES THEM IN MEMORY.
                        ;
0286 217410     BINH:   LXI     H,HCON  ;CONVERSION
0289 47                 MOV     B,A     ;SAVE VALUE
028A 1F                 RAR
028B 1F                 RAR
028C 1F                 RAR
028D 1F                 RAR
028E CD9902             CALL    BIN1
0291 77                 MOV     M,A
0292 23                 INX     H
0293 78                 MOV     A,B
0294 CD9902             CALL    BIN1    ;CONVERT TO ASCII
0297 77                 MOV     M,A
0298 C9                 RET
                        ;
                        ; THIS ROUTINE CONVERTS A VALUE TO HEXADECIMAL
                        ;
0299 E60F       BIN1:   ANI     0FH     ;LOW 4 BITS
029B C630               ADI     48      ;CONVERT TO ASCII
029D FE3A               CPI     58      ;DIGIT 0-9
029F D8                 RC
02A0 C607               ADI     7       ;MODIFY FOR A-F
02A2 C9                 RET
                        ;
                        ;
                        ; THIS ROUTINE CONVERTS THE BINARY VALUE IN REG A INTO
                        ; ASCII DECIMAL DIGITS AND STORES THEM IN MEMORY
                        ;
02A3 217410     BIND:   LXI     H,HCON  ;CONVERSION ADDRESS
02A6 0664               MVI     B,100
02A8 CDB402             CALL    BID1    ;CONVERT HUNDREDS DIGIT
02AB 060A               MVI     B,10
02AD CDB402             CALL    BID1    ;CONVERT TENS DIGIT
02B0 C630               ADI     '0'     ;GET UNITS DIGIT
02B2 77                 MOV     M,A     ;STORE IN MEMORY
02B3 C9                 RET
                        ;
                        ; THIS ROUTINE CONVERTS A VALUE TO DECIMAL
                        ;
02B4 362F       BID1:   MVI     M,'0'-1 ;INITIALIZE DIGIT COUNT
02B6 34                 INR     M
02B7 90                 SUB     B       ;CHECK DIGIT
02B8 D2B602             JNC     BID1+2
02BB 80                 ADD     B       ;RESTORE VALUE
02BC 23                 INX     H
02BD C9                 RET
                        ;
                        ;
                        ; LEGAL COMMAND TABLE
                        ;
02BE 44554D50   CTAB:   DB      'DUMP'  ;DUMP COMMAND
02C2 0803               DW      DUMP    :COMMAND ADDRESS
```

```
02C4 45584543          DB      'EXEC'  ;EXECUTE COMMAND
02C8 2101              DW      EXEC    ;COMMAND ADDRESS
02CA 454E5452          DB      'ENTR'  ;ENTER COMMAND
02CE 7604              DW      ENTR
02D0 46494C45          DB      'FILE'  ;FILE COMMAND
02D4 3E03              DW      FILE    ;COMMAND ADDRESS
02D6 4C495354          DB      'LIST'  ;LIST COMMAND
02DA D005              DW      LIST    ;COMMAND ADDRESS
02DC 44454C54          DB      'DELT'  ;DELETE COMMAND
02E0 E705              DW      DELL    ;COMMAND ADDRESS
02E2 4153534D          DB      'ASSM'  ;ASSEMBLE COMMAND
02E6 5E06              DW      ASSM    ;COMMAND ADDRESS
02E8 50414745          DB      'PAGE'  ;PAGE TRANSFER COMMAND
02EC 2203              DW      PAGE    ;COMMAND ADDRESS
02EE 43555354          DB      'CUST'  ;CUSTOMER COMMAND
02F2 0020              DW      2000H   ;COMMAND ADDRESS
02F4 4252454B          DB      'BREK'  ;BREAKPOINT COMMAND
02F8 D20C              DW      BREAK   ;COMMAND ADDRESS
02FA 50524F43          DB      'PROC'  ;;PROCEED COMMAND
02FE 8F0D              DW      PROC    ;COMMAND ADDRESS
                       ;
                       ;
                       ; THIS ROUTINE CHECKS IF ANY PARAMETERS WERE ENTERED
                       ; WITH THE COMMAND, IF NOT AN ERROR MESSAGE IS ISSUED
                       ;
0300 3A7E10    VCHK:   LDA     ABUF    ;FETCH PARAMETER BYTE
0303 B7                ORA     A       ;SET FLAGS
0304 CA5A04            JZ      WHAT    ;NO PARAMETER
0307 C9                RET
                       ;
                       ;
                       ; THIS ROUTINE DUMPS OUT THE CONTENTS OF MEMORY FROM
                       ; THE START TO FINAL ADDRESSES GIVEN IN THE COMMAND.
                       ;
0308 CD0003    DUMP:   CALL    VCHK    ;CHECK FOR PARAMETERS
030B CD0E01    DUMS:   CALL    CRLF    ;START NEW LINE
030E 2A8A10    DUM1:   LHLD    BBUF    ;FETCH MEMORY ADDRESS
0311 7E                MOV     A,M
0312 CD4A02            CALL    HQTB    ;OUTPUT VALUE
0315 CD6302            CALL    ACHK    ;CHECK ADDRESS
0318 D8                RC              ;RETURN IF FINISHED
0319 7D                MOV     A,L     ;IS NEXT ADDRESS
031A E60F              ANI     0FH     ;  DIVISIBLE BY 16?
031C C20E03            JNZ     DUM1
031F C30B03            JMP     DUMS
                       ;
                       ;
                       ; THIS ROUTINE WILL MOVE 256 BYTES FROM 1ST ADDRESS
                       ; GIVEN IN COMMAND TO 2ND ADDRESS IN COMMAND.
                       ;
0322 CD0003    PAGE:   CALL    VCHK    ;CHECK FOR PARAMETER
0325 3A8210            LDA     ABUF+4  ;FETCH 2ND PARAMETER
0328 B7                ORA     A       ;DOES 2ND PARAMETER EXIST?
0329 CA5A04            JZ      WHAT
032C 2A8A10            LHLD    BBUF    ;FETCH MOVE TO ADDRESS
032F EB                XCHG
0330 2A8C10            LHLD    BBUF+2  ;FETCH MOVE TO ADDRESS
0333 0600              MVI     B,0     ;SET COUNTER
0335 1A        PAG1:   LDAX    D
0336 77                MOV     M,A
0337 23                INX     H
0338 13                INX     D
0339 05                DCR     B       ;DECREMENT COUNT
033A C23503            JNZ     PAG1
033D C9                RET
                       ;
```

```
                        ;
                        ;
                        ; THIS ROUTINE INITIALIZES THE BEGINNING OF FILE ADDRESS
                        ; AND END OF FILE ADDRESS AS WELL AS THE FILE AREA
                        ; WHEN THE FILE COMMAND IS USED
                        ;
033E CD0E01     FILE:   CALL    CRLF
                        ; CHECK FOR FILE PARAMETERS
0341 3A7610             LDA     FBUF
0344 B7                 ORA     A
0345 CAB903             JZ      FOUT    ;NO - GO LIST
0348 CD1804             CALL    FSEA    ;LOOK UP FILE
034B EB                 XCHG            ;PNTR IN DE
034C C26303             JNZ     TEST
                        ; NO ENTRY
034F 3A7E10             LDA     ABUF    ;CHECK FOR PARAM
0352 B7                 ORA     A
0353 CA5D04             JZ      WHA1    ;NO?? - ERROR
                        ; CHECK FOR ROOM IN DIRECTORY
0356 3A7D10             LDA     FEF
0359 B7                 ORA     A
035A C27803             JNZ     ROOM
035D 216B04             LXI     H,EMES1
0360 C36004             JMP     MESS
                        ; ENTRY FOUND ARE THESE PARAMETERS
0363 3A7E10     TEST:   LDA     ABUF
0366 B7                 ORA     A
0367 CA8B03             JZ      SWAPS
036A 2A8A10             LHLD    BBUF
036D 7C                 MOV     A,H
036E B5                 ORA     L
036F CA8B03             JZ      SWAPS
0372 217004             LXI     H,EMES2 ;NO-NO CAN'T DO
0375 C36004             JMP     MESS    ;IT - DELETE FIRST
                        ; MOVE FILE NAME TO BLOCK POINTED TO BY FREAD
0378 2A7B10     ROOM:   LHLD    FREAD
037B EB                 XCHG
037C 217610             LXI     H,FBUF  ;FILE NAME POINTER IN H,L
037F D5                 PUSH    D
0380 0E05               MVI     C,NMLEN ;NAME LENGTH COUNT
0382 7E         MOV23:  MOV     A,M
0383 12                 STAX    D
0384 13                 INX     D
0385 0D                 DCR     C       ;TEST COUNT
0386 23                 INX     H
0387 C28203             JNZ     MOV23
038A D1                 POP     D       ;RESTORE ENTRY POINTER
                        ; MAKE FILE POINTED TO BY D,E CURRENT
038B 212410     SWAPS:  LXI     H,FILE0
038E 0E0D               MVI     C,FELEN ;ENTRY LENGTH
0390 1A         SWAP:   LDAX    D
0391 46                 MOV     B,M
0392 77                 MOV     M,A     ;EXCHANGE
0393 78                 MOV     A,B
0394 12                 STAX    D
0395 13                 INX     D
0396 23                 INX     H       ;BUMP POINTERS
0397 0D                 DCR     C       ;TEST COUNT
0398 C29003             JNZ     SWAP
                        ; CHECK FOR 2ND PARAMETER
039B 3A7E10             LDA     ABUF
039E B7                 ORA     A
039F CAC303             JZ      FOOT    ;NO SECOND PARAMETER
                        ; PROCESS 2ND PARAMETER
03A2 2A8A10             LHLD    BBUF    ;GET ADDRESS
03A5 222910             SHLD    BOFP    ;SET BEGIN
```

```
03A8 222B10             SHLD    EOFP    ;SET END
03AB 7D                 MOV     A,L     ;IS ADDRESS ZERO?
03AC B4                 ORA     H
03AD CAB203             JZ      FIL35   ;YES
03B0 3601       FIL30:  MVI     M,1     ;NON-ZERO - SET EOF
03B2 AF         FIL35:  XRA     A       ;AND MAX LINE #
03B3 322D10             STA     MAXL
03B6 C3C303             JMP     FOOT    ;OUTPUT PARAMETERS
03B9 3ACB10     FOUT:   LDA     IBUF+4
03BC FE53               CPI     'S'     ;IS COMMAND FILES?
03BE 0E06               MVI     C,MAXFIL
03C0 CAC503             JZ      FOUL
03C3 0E01       FOOT:   MVI     C,1
                ; OUTPUT THE # OF ENTRIES IN C
03C5 212410     FOUL:   LXI     H,FILE0
03C8 79                 MOV     A,C
03C9 327D10     FINE:   STA     FOCNT   ;SAVE COUNT
03CC E5                 PUSH    H
03CD 110500             LXI     D,NMLEN
03D0 19                 DAD     D
03D1 7E                 MOV     A,M
03D2 B7                 ORA     A
03D3 C2E303             JNZ     FOOD    ;NON ZERO, OK TO OUTPUT
03D6 23                 INX     H
03D7 86                 ADD     M
03D8 23                 INX     H
03D9 C2E303             JNZ     FOOD
03DC 33                 INX     SP
03DD 33                 INX     SP
03DE 23                 INX     H
03DF 23                 INX     H
03E0 C3F803             JMP     FEET
                ; HAVE AN ENTRY TO OUTPUT
03E3 E1         FOOD:   POP     H       ;PTR
03E4 0E05               MVI     C,NMLEN
03E6 46         FAST:   MOV     B,M     ;LOAD CHARACTER TO B
03E7 CD0301             CALL    OUT8
03EA 0D                 DCR     C
03EB 23                 INX     H
03EC C2E603             JNZ     FAST    ;DO THE REST
                ; NOW OUTPUT BEGIN-END PTRS
03EF CD0404             CALL    FOOL    ;OUTPUT BEGIN
03F2 CD0404             CALL    FOOL    ;OUTPUT END
03F5 CD0E01             CALL    CRLF    ;AND C/R
                ; TEST COUNT, H,L POINTS PAST EOFP
03F8 110400     FEET:   LXI     D,FELEN-NMLEN-4
03FB 19                 DAD     D       ;MOVE TO NEXT ENTRY
03FC 3A7D10             LDA     FOCNT
03FF 3D                 DCR     A       ;TEST COUNT
0400 C2C903             JNZ     FINE    ;MORE TO DO
0403 C9                 RET             ;DONE!
                ; OUTPUT NUMBER POINTED TO BY H,L
                ; ON RET, H,L POINT 2 WORDS LATER
0404 CD5D02     FOOL:   CALL    BLK1    ;SPACE
0407 23                 INX     H
0408 7E                 MOV     A,M
0409 2B                 DCX     H
040A E5                 PUSH    H
040B CD3A02             CALL    HOUT    ;OUTPUT
040E E1                 POP     H
040F 7E                 MOV     A,M
0410 23                 INX     H
0411 23                 INX     H
0412 E5                 PUSH    H
0413 CD4A02             CALL    HOTB    ;OUTPUT
0416 E1                 POP     H       ;RESTORE H,L
```

```
0417 C9                 RET
                ; SEARCH THE FILE DIRECTORY FOR THE FILE
                ; WHOSE NAME IS IN FBUF.
                ; RETURN IF FOUND, ZERO IS OFF, H,L POINT TO
                ; ENTRY WHILE SEARCHING, ON ENTRY FOUND WITH ADDR
                ; ZERO, SET FEF TO >0 AND FREAD TO THE ADDR OF ENTRY
                ;
0418 AF         FSEA:   XRA     A
0419 327D10             STA     FEF     ;CLAIM NO FREE ENTRIES
041C 0606               MVI     B,MAXFIL        ;COUNT OF ENTRIES
041E 112410             LXI     D,FILE0 ;TABLE ADDRESS
0421 217610     FSE10:  LXI     H,FBUF
0424 0E05               MVI     C,NMLEN
0426 CD5301             CALL    SEAR    ;TEST STRINGS
0429 F5                 PUSH    PSW     ;SAVE FLAG
042A D5                 PUSH    D
042B 1A                 LDAX    D       ;GET BOFP
042C B7                 ORA     A       ;EMPTY ENTRY?
042D C24E04             JNZ     FSE20
0430 13                 INX     D       ;STORE OTHER WORD
0431 1A                 LDAX    D
0432 B7                 ORA     A
0433 C24E04             JNZ     FSE20   ;NOPE-GO TEST FOR MATCH
0436 EB                 XCHG
0437 11FAFF             LXI     D,-NMLEN-1
043A 19                 DAD     D       ;MOV TO BEGINNING
043B 227B10             SHLD    FREAD   ;SAVE ADDR
043E 7A                 MOV     A,D
043F 327D10             STA     FEF     ;SET FREE ENTRY FOUND
0442 E1                 POP     H       ;RESTOR INTERIM PTR
0443 F1                 POP     PSW     ;UNJUNK STACK
                ; MOVE TO NEXT ENTRY
0444 110800     FSE15:  LXI     D,FELEN-NMLEN
0447 19                 DAD     D
0448 EB                 XCHG            ;NEXT ENTRY ADDR IN DE
0449 05                 DCR     B       ;TEST COUNT
044A C8                 RZ              ;DONE--NOPE
044B C32104             JMP     FSE10   ;TRY NEXT
                ;ENTRY WASN'T FREE, TEST FOR MATCH
044E E1         FSE20:  POP     H
044F F1                 POP     PSW
0450 C24404             JNZ     FSE15   ;IF ZERO CLEAR, NO MATCH
                ;ENTRY FOUND
0453 11FBFF             LXI     D,-NMLEN        ;BACKUP
0456 19                 DAD     D       ;H,L POINTS TO ENTRY
0457 7A                 MOV     A,D
0458 B7                 ORA     A       ;CLEAR ZERO
0459 C9                 RET             ;THAT'S ALL
                ;
                ;
                ; OUTPUT ERROR MESSAGE FOR ILLEGAL COMMAND
                ;
045A CD0E01     WHAT:   CALL    CRLF    ;OUT CRLF
045D 216604     WHA1:   LXI     H,EMES  ;MESSAGE ADDRESS
0460 CD7A02     MESS:   CALL    SCRN
0463 C36700             JMP     EOR
                ;
0466 57484154   EMES:   DB      'WHAT'
046A 0D                 DB      13      ;CARRIAGE RETURN
046B 46554C4C   EMES1:  DB      'FULL',13
046F 0D
0470 4E4F204E   EMES2:  DB      'NO NO',13
0474 4F0D
                ;
                ;
                ; CALL ROUTINE TO ENTER DATA INTO MEMORY
```

```
                        ;
                        ; AND CHECK FOR ERROR ON RETURN
                        ;
                        ; THIS ROUTINE IS USED TO ENTER DATA VALUES INTO MEMORY.
                        ; EACH VALUE IS ONE BYTE AND IS WRITTEN IN HEXADECIMAL
                        ; VALUES GREATER THAT 255 WILL CAUSE CARRY TO BE SET
                        ; AND RETURN TO BE MADE TO CALLING PROGRAM
                        ;
0476 CD0003     ENTR:   CALL    VCHK    ;CHECK FOR PARAMETERS
0479 CD8304             CALL    ENTS
047C DA5A04             JC      WHAT
047F CD0E01             CALL    CRLF
0482 C9                 RET
                        ;
                        ;
002F            EEND    EQU     '/'     ;TERMINATION CHAR
0483 CD0E01     ENTS:   CALL    CRLF
0486 CD8000             CALL    READ    ;READ INPUT DATA
0489 21C710             LXI     H,IBUF  ;SET LINE POINTER
048C 229610             SHLD    PNTR    ;SAVE POINTER
048F CD6601     ENT1:   CALL    ZBUF    ;CLEAR BUFFER
0492 CD0D09             CALL    SBLK    ;CAN TO FIRST VALUE
0495 DA8304             JC      ENTS    ;JUMP IF CR FOUND
0498 FE2F               CPI     EEND
049A C8                 RZ              ;RETURN CARRY IS ZERO
049B CD7508             CALL    ALPS    ;PLACE VALUE IN BUFFER
049E 78                 MOV     A,B     ;GET DIGIT COUNT
049F FE03               CPI     3       ;CHECK NUR OF DIGITS
04A1 3F                 CMC
04A2 D8                 RC              ;RETURN IF MORE THAN    2 DIGITS
04A3 017E10             LXI     B,ABUF  ;CONVERSION ADDRESS
04A6 CD1B02             CALL    AHEX    ;CONVERT VALUE
04A9 D8                 RC              ;ERROR IN HEX CHARACTER
04AA 7D                 MOV     A,L
04AB 2A8A10             LHLD    BBUF    ;FETCH MEMORY ADDRESS
04AE 77                 MOV     M,A     ;PUT IN MEMORY
04AF CD7502             CALL    ACH1    ;INCREMENT MEMORY LOCATION
04B2 C38F04             JMP     ENT1
                        ;
                        ;
                        ;
                        ; THIS ROUTINE IS USED TO ENTER LINES INTO THE FILE
                        ; AREA. THE LINE NUMBER IS FIRST CHECKED TO SEE IF IT IS
                        ; A VALID NUMBER (0000-9999). NEXT IT IS CHECKED TO SEE
                        ; IF IT IS GREATER THAN THE MAXIMUM CURRENT LINE NUMBER.
                        ; IF IT IS, THE NEXT LINE IS INSERTED AT THE END OF THE
                        ; CURRENT FILE AND THE MAXIMUM LINE NUMBER IS UPDATED AS
                        ; WELL AS THE END OF FILE POSITION. LINE NUMBERS THAT
                        ; ALREADY EXIST ARE INSERTED INTO THE FILE AREA AT THE
                        ; APPROPRIATE PLACE AND ANY EXTRA CHARACTERS IN THE OLD
                        ; LINE ARE DELETED.
                        ;
04B5 3A2410     LINE:   LDA     FILE0   ;IS A FILE DEFINED?...
04B8 B7                 ORA     A
04B9 CA5A04             JZ      WHAT    ;ABORT IF NOT
04BC 0E04               MVI     C,4     ;NO OF DIGITS TO CHECK
04BE 21C610             LXI     H,IBUF-1        ;INITIALIZE ADDRESS
04C1 23         LICK:   INX     H
04C2 7E                 MOV     A,M     ;FETCH LINE DIGIT
04C3 FE30               CPI     '0'     ;CHECK FOR VALID NUMBER
04C5 DA5A04             JC      WHAT
04C8 FE3A               CPI     '9'+1
04CA D25A04             JNC     WHAT
04CD 0D                 DCR     C
04CE C2C104             JNZ     LICK
04D1 227410             SHLD    ADDS    ;FIND ADDRESS
04D4 113010             LXI     D.MAXL+3        :GET ADDRESS
```

```
04D7 CDA205             CALL    COM0
04DA D2FA04             JNC     INSR
                ; GET HERE IF NEW LINE IS GREATER THAN MAXIMUM LINE #
04DD 23                 INX     H
04DE CD9205             CALL    LODM    ;GET NEW LINE NUMBER
04E1 213010             LXI     H,MAXL+3
04E4 CD9A05             CALL    STOM    ;MAKE IT MAXIMUM LINE NUMBER
04E7 11C610             LXI     D,IBUF-1
04EA 2A2B10             LHLD    EOFP    ;END OF FILE POSITION
04ED 0E01               MVI     C,1
04EF CD8005             CALL    LMOV    ;PLACE LINE IN FILE
04F2 3601       SEOF:   MVI     M,1     ;END OF FILE INDICATOR
04F4 222B10             SHLD    EOFP    ;END OF FILE ADDRESS
04F7 C36700             JMP     EOR
                ; GET HERE IF NEW LINE MUST BE INSERTED INTO ALREADY
                ; EXISTING FILE AREA
04FA CD5205     INSR:   CALL    FIN1    ;FIND LINE IN FILE
04FD 0E02               MVI     C,2
04FF CA0305             JZ      EQUL
0502 0D                 DCR     C       ;NEW LN NOT EQUAL TO SOME OLD LN
0503 46         EQUL:   MOV     B,M
0504 2B                 DCX     H
0505 3602               MVI     M,2     ;MOVE LINE INDICATOR
0507 227210             SHLD    INSP    ;INSERT LINE POSITION
050A 3AC610             LDA     IBUF-1  ;NEW LN COUNT
050D 0D                 DCR     C
050E CA1805             JZ      LT      ;NEW LN NOT = OLD LN
0511 90                 SUB     B       ;COUNT DIFFERENCE
0512 CA3B05             JZ      ZERO    ;LINE LENGTHS EQUAL
0515 DA2B05             JC      GT
                ; GET HERE IF # OF CHARS IN OLD LINE > # OF CHARS IN
                ; NEW LINE OR NEW LINE # WAS NOT EQUAL TO SOME OLD
                ; LINE #
0518 2A2B10     LT:     LHLD    EOFP    ;END OF FILE ADDRESS
051B 54                 MOV     D,H
051C 5D                 MOV     E,L
051D CD7B05             CALL    ADR     ;MOVE TO ADDRESS
0520 222B10             SHLD    EOFP    ;NEW END OF FILE ADDRESS
0523 0E02               MVI     C,2
0525 CD8905             CALL    RMOV    ;OPEN UP FILE AREA
0528 C33B05             JMP     ZERO
                ; GET HERE IF # OF CHARS IN OLD LINE < # OF CHARS IN
                ; NEW LINE.
052B 2F         GT:     CMA
052C 3C                 INR     A       ;COUNT DIFFERENCE
052D 54                 MOV     D,H
052E 5D                 MOV     E,L
052F CD7B05             CALL    ADR
0532 EB                 XCHG
0533 CD8005             CALL    LMOV    ;DELETE EXCESS CHAR IN FILE
0536 3601               MVI     M,1     ;E-O-F INDICATOR
0538 222B10             SHLD    EOFP    ;E-O-F ADDRESS
                ; GET HERE TO INSERT CURRENT LINE INTO FILE AREA
053B 2A7210     ZERO:   LHLD    INSP    ;INSERT ADDRESS
053E 360D               MVI     M,ASCR
0540 23                 INX     H
0541 11C610             LXI     D,IBUF-1        ;NEW LINE ADDRESS
0544 0E01               MVI     C,1     ;CHECK VALUE
0546 CD8005             CALL    LMOV    ;PLACE LINE IN FILE
0549 C36700             JMP     EOR
                ;
                ;
                ; THIS ROUTINE IS USED TO FIND A LN IN THE FILE AREA
                ; WHICH IS GREATER THAN OR EQUAL TO THE CURRENT LINE #
054C 218110     FIND:   LXI     H,ABUF+3        ;BUFFER ADDRESS
054F 227410             SHLD    ADDS    :SAVE ADDRESS
```

```
0552 2A2910    FIN1:   LHLD    BOFP    ;BEGIN FILE ADDRESS
0555 7C                MOV     A,H     ;RETURN TO MONITOR IF
0556 B5                ORA     L       ;  FILE IS EMPTY...
0557 CA6700            JZ      EOR
055A CD7405    FI1:    CALL    EO1     ;CHECK FOR END OF FILE
055D EB                XCHG
055E 2A7410            LHLD    ADDS    ;FETCH FIND ADDRESS
0561 EB                XCHG
0562 3E04              MVI     A,4
0564 CD7B05            CALL    ADR     ;BUMP LINE ADDRESS
0567 CDA205            CALL    COM0    ;COMPARE LINE NUMBERS
056A D8                RC
056B C8                RZ
056C 7E        FI2:    MOV     A,M
056D CD7B05            CALL    ADR     ;NEXT LINE ADDRESS
0570 C35A05            JMP     FI1
               ;
               ;
               ; WHEN SEARCHING THROUGH THE FILE AREA, THIS ROUTINE
               ; CHECKS TO SEE IF THE CURRENT ADDRESS IS THE END OF
               ; FILE
               ;
0573 23        EOF:    INX     H
0574 3E01      EO1:    MVI     A,1     ;E-O-F INDICATOR
0576 BE                CMP     M
0577 C0                RNZ
0578 C36700            JMP     EOR
               ;
               ;
               ; THIS ROUTINE IS USED TO ADD A VALUE TO AN ADDRESS
               ; CONTAINED IN REGISTER H,L
               ;
057B 85        ADR:    ADD     L
057C 6F                MOV     L,A
057D D0                RNC
057E 24                INR     H
057F C9                RET
               ;
               ;
               ; THIS ROUTINE WILL MOVE CHARACTER STRINGS FROM ONE
               ; LOCATION OF MEMORY TO ANOTHER
               ; CHARACTERS ARE MOVED FROM LOCATION ADDRESSED BY D,E
               ; TO LOCATION ADDRESSED BY H,L.  ADDITIONAL CHARACTERS
               ; ARE MOVED BY BUMPING POINTERS UNTIL THE CHARACTER IN
               ; REG C IS FETCHED.
               ;
0580 1A        LMOV:   LDAX    D       ;FETCH CHARACTER
0581 13                INX     D       ;INCREMENT FETCH ADDRESS
0582 B9                CMP     C       ;TERMINATION CHARACTER
0583 C8                RZ
0584 77                MOV     M,A     ;STORE CHARACTER
0585 23                INX     H       ;INCREMENT STORE ADDRESS
0586 C38005            JMP     LMOV
               ;
               ;
               ; THIS ROUTINE IS SIMILAR TO ABOVE EXCEPT THAT THE
               ; CHARACTER ADDRESS IS DECREMENTED AFTER EACH FETCH
               ; AND STORE
               ;
0589 1A        RMOV:   LDAX    D       ;FETCH CHARACTER
058A 1B                DCX     D       ;DECREMENT FETCH ADDRESS
058B B9                CMP     C       ;TERMINATION CHARACTER
058C C8                RZ
058D 77                MOV     M,A     ;STORE CHARACTER
058E 2B                DCX     H       ;DECREMENT STORE ADDRESS
058F C38905            JMP     RMOV
```

```
                        ;
                        ;
                        ; THIS ROUTINE IS USED TO LOAD FOUR CHARACTERS FROM
                        ; MEMORY INTO REGISTERS
                        ;
0592 46         LODM:   MOV     B,M     ;FETCH CHARACTER
0593 23                 INX     H
0594 4E                 MOV     C,M     ;FETCH CHARACTER
0595 23                 INX     H
0596 56                 MOV     D,M     ;FETCH CHARACTER
0597 23                 INX     H
0598 5E                 MOV     E,M     ;FETCH CHARACTER
0599 C9                 RET
                        ;
                        ;
                        ; THIS ROUTINE STORES FOUR CHARACTERS FROM THE REGISTERS
                        ; INTO MEMORY
                        ;
059A 73         STOM:   MOV     M,E     ;STORE CHARACTER
059B 2B                 DCX     H
059C 72                 MOV     M,D     ;STORE CHARACTER
059D 2B                 DCX     H
059E 71                 MOV     M,C     ;STORE CHARACTER
059F 2B                 DCX     H
05A0 70                 MOV     M,B     ;STORE CHARACTER
05A1 C9                 RET
                        ;
                        ;
                        ; THIS ROUTINE IS USED TO COMPARE TWO CHARACTER STRINGS
                        ; OF LENGTH 4, ON RETURN ZERO FLAG SET MEANS BOTH
                        ; STRINGS ARE EQUAL.  CARRY FLAG =0 MEANS STRING ADDRESS
                        ; BY D,E WAS GREATER THAN OR EQUAL TO CHARACTER STRING
                        ; ADDRESSED BY H,L
                        ;
05A2 0601       COM0:   MVI     B,1     ;EQUAL COUNTER
05A4 0E04               MVI     C,4     ;STRING LENGTH
05A6 B7                 ORA     A       ;CLEAR CARRY
05A7 1A         CO1:    LDAX    D       ;FETCH CHARACTER
05A8 9E                 SBB     M       ;COMPARE CHARACTERS
05A9 CAAD05             JZ      CO2
05AC 04                 INR     B       ;INCREMENT EQUAL COUNTER
05AD 1B         CO2:    DCX     D
05AE 2B                 DCX     H
05AF 0D                 DCR     C
05B0 C2A705             JNZ     CO1
05B3 05                 DCR     B
05B4 C9                 RET
                        ;
                        ; THIS ROUTINE IS SIMILAR TO THE ABOVE ROUTINE EXCEPT ON
                        ; RETURN CARRY FLAG = 0 MEANS THAT CHARACTER STRING
                        ; ADDRESSED BY D,E IS ONLY > STRING ADDRESSED BY H,L.
                        ;
05B5 0E04       COM1:   MVI     C,4     ;STRING LENGTH
05B7 1A                 LDAX    D       ;TCH CHARACTER
05B8 D601               SUI     1
05BA C3A805             JMP     CO1+1
                        ;
                        ;
                        ; THIS ROUTINE WILL TAKE ASCII CHARACTERS AND ADD ANY
                        ; NECESSARY ASCII ZEROES SO THE RESULT IS A 4 CHARACTER
                        ; ASCII VALUE
                        ;
05BD CD9205     NORM:   CALL    LODM    ;LOAD CHARACTERS
05C0 AF                 XRA     A       ;FETCH A ZERO
05C1 B8                 CMP     B
05C2 C8                 RZ
```

```
05C3 B8        NOR1:   CMP     E
05C4 C49A05            CNZ     STOM    ;STORE VALUES
05C7 C0                RNZ
05C8 5A                MOV     E,D     ;NORMALIZE VALUE
05C9 51                MOV     D,C
05CA 48                MOV     C,B
05CB 0630              MVI     B,'0'
05CD C3C305            JMP     NOR1
                ;
                ;
                ; THIS ROUTINE IS USED TO LIST THE CONTENTS OF THE FILE
                ; AREA STARTING AT THE LINE NUMBER GIVEN IN THE COMMAND
                ;
05D0 CD0E01    LIST:   CALL    CRLF
05D3 CD4C05            CALL    FIND    ;FIND STARTING LN
05D6 23        LIST0:  INX     H       ;OUTPUT LINE...
05D7 CD7A02            CALL    SCRN
05DA CD0E01            CALL    CRLF
05DD CD7305            CALL    EOF     ;CHECK FOR END OF FILE
05E0 CDE900            CALL    INK     ;CHECK FOR ^X
05E3 C2D605            JNZ     LIST0   ;LOOP IF NO ^X
05E6 C9                RET
                ;
                ;
                ; THIS ROUTINE IS USED TO DELETE LINES FROM THE
                ; FILE AREA.  THE REMAINING FILE AREA IS THEN MOVED IN
                ; MEMORY SO THAT THERE IS NO EXCESS SPACE.
                ;
05E7 CD0003    DELL:   CALL    VCHK    ;CHECK FOR PARAMETER
05EA CD4C05            CALL    FIND    ;FIND LINE IN FILE AREA
05ED 227210            SHLD    DELP    ;SAVE DELETE POSITION
05F0 218510            LXI     H,ABUF+7
05F3 7E                MOV     A,M     ;CHECK FOR 2ND PARAMETER
05F4 B7                ORA     A       ;SET FLAGS
05F5 C2FB05            JNZ     DEL1
05F8 218110            LXI     H,ABUF+3        ;USE FIRST PARAMETER
05FB 227410    DEL1:   SHLD    ADDS    ;SAVE FIND ADDRESS
05FE EB                XCHG
05FF 213010            LXI     H,MAXL+3
0602 CDA205            CALL    COM0    ;COMPARE LINE NUMBERS
0605 2A7210            LHLD    DELP    ;LOAD DELETE POSITION
0608 DA4906            JC      NOVR
                ; GET HERE IF DELETION INVOLVES END OF FILE
060B 222B10            SHLD    EOFP    ;CHANGE E-O-F POSITION
060E 3601              MVI     M,1     ;SET E-O-F INDICATOR
0610 EB                XCHG
0611 2A2910            LHLD    BOFP    ;GET BEGIN FILE ADDRESS
0614 EB                XCHG
0615 060D              MVI     B,13    ;SET SCAN SWITCH
0617 2B                DCX     H       ;CHECK FOR BOF
0618 7D        DEL2:   MOV     A,L
0619 93                SUB     E
061A 7C                MOV     A,H
061B 9A                SBB     D
061C 3E0D              MVI     A,ASCR  ;LOOK FOR CR
061E DA4006            JC      DEL4    ;DECREMENTED PAST BOF
0621 05                DCR     B
0622 2B                DCX     H
0623 BE                CMP     M       ;FIND NEW MAX LN
0624 C21806            JNZ     DEL2
0627 2B                DCX     H
0628 7D                MOV     A,L
0629 93                SUB     E
062A 7C                MOV     A,H
062B 9A                SBB     D
062C DA4106            JC      DEL5
```

```
062F BE              CMP     M       ;END OF PREVIOUS LINE
0630 23              INX     H
0631 23              INX     H
0632 CA3606          JZ      DEL3
0635 23              INX     H
0636 CD9205  DEL3:   CALL    LODM    ;LOAD NEW MAX LN
0639 213010          LXI     H,MAXL+3        ;SET ADDRESS
063C CD9A05          CALL    STOM    ;STORE NEW MAX LN
063F C9              RET
0640 B8      DEL4:   CMP     B       ;CHECK SWITCH
0641 EB      DEL5:   XCHG
0642 C23506          JNZ     DEL3-1
0645 322D10          STA     MAXL    ;MAKE MAX LN A SMALL NUMBER
0648 C9              RET
             ; GET HERE IF DELETION IS IN MIDDLE OF FILE AREA
0649 CD5A05  NOVR:   CALL    FI1     ;FIND END OF DELETE AREA
064C CC6C05          CZ      FI2     ;NEXT LINE IF THIS LN EQUAL
064F EB      NOV1:   XCHG
0650 2A7210          LHLD    DELP    ;CHAR MOVE TO POSITION
0653 0E01            MVI     C,1     ;MOVE TERMINATOR
0655 CD8005          CALL    LMOV    ;COMPACT FILE AREA
0658 222B10          SHLD    EOFP    ;SET EOF POSITION
065B 3601            MVI     M,1     ;SET EOF INDICATOR
065D C9              RET
                     ;
                     ; STARTING HERE IS THE SELF ASSEMBLER PROGRAM
                     ; THIS PROGRAM ASSEMBLES PROGRAMS WHICH ARE
                     ; IN THE FILE AREA
                     ;
065E CD0003  ASSM:   CALL    VCHK    ;CHECK FOR PARAMETER
0661 3A8210          LDA     ABUF+4  ;GET 2ND PARAMETER
0664 B7              ORA     A       ;CHECK FOR PARAMETERS
0665 C26E06          JNZ     ASM4
0668 2A8A10          LHLD    BBUF    ;FETCH 1ST PARAMETER
066B 228C10          SHLD    BBUF+2  ;STORE INTO 2ND PARAMETER
066E 3ACB10  ASM4:   LDA     IBUF+4  ;FETCH INPUT CHARACTER
0671 D645            SUI     'E'     ;RESET A IF ERRORS ONLY
0673 328E10          STA     AERR    ;SAVE ERROR FLAG
0676 AF              XRA     A       ;GET A ZERO
0677 329810          STA     NOLA    ;INITIALIZE LABEL COUNT
067A 329410  ASM3:   STA     PASI    ;SET PASS INDICATOR
067D CD0E01          CALL    CRLF    ;INDICATE START OF PASS
0680 2A8A10          LHLD    BBUF    ;FETCH ORIGIN
0683 229210          SHLD    ASPC    ;INITIALIZE PC
0686 2A2910          LHLD    BOFP    ;GET START OF FILE
0689 227210          SHLD    APNT
068C 2A7210  ASM1:   LHLD    APNT    ;FETCH LINE POINTER
068F 31B210          LXI     SP,AREA+18
0692 7E              MOV     A,M     ;FETCH CHARACTER
0693 FE01            CPI     1       ;END OF FILE?
0695 CA0109          JZ      EASS    ;JUMP IF END OF FILE
0698 EB              XCHG
0699 13              INX     D       ;INCREMENT ADDRESS
069A 21B210          LXI     H,OBUF  ;BLANK START ADDRESS
069D 3EC2            MVI     A,IBUF-5 AND 0FFH ;BLANK END ADDRESS
069F CDE100          CALL    CLER    ;BLANK OUT BUFFER
06A2 0E0D            MVI     C,ASCR  ;STOP CHARACTER
06A4 CD8005          CALL    LMOV    ;MOVE LINE INTO BUFFER
06A7 71              MOV     M,C     ;PLACE CR IN BUFFER
06A8 EB              XCHG
06A9 227210          SHLD    APNT    ;SAVE ADDRESS
06AC 3A9410          LDA     PASI    ;FETCH PASS INDICATOR
06AF B7              ORA     A       ;SET FLAGW
06B0 C2B906          JNZ     ASM2    ;JUMP IF PASS 2
06B3 CDDC06          CALL    PASI
06B6 C38C06          JMP     ASM1
```

```
06B9 CD9307    ASM2:   CALL    PAS2
06BC 21B210            LXI     H,OBUF  ;OUTPUT BUFFER ADDRESS
06BF CDC506            CALL    AOUT    ;OUTPUT LINE
06C2 C38C06            JMP     ASM1
                       ;
                       ; THIS ROUTINE IS USED TO OUTPUT THE LISTING FOR
                       ; AN ASSEMBLY.  IT CHECKS THE ERROR SWITCH TO SEE IF
                       ; ALL LINES ARE TO BE PRINTED OR JUST THOSE WITH
                       ; ERRORS.
                       ;
06C5 3A8E10    AOUT:   LDA     AERR    ;FETCH ERROR SWITCH
06C8 B7                ORA     A       ;SET FLAGS
06C9 C2D206            JNZ     AOU1    ;OUTPUT ALL LINES
06CC 3AB210    AOU2:   LDA     OBUF    ;FETCH ERROR INDICATOR
06CF FE20              CPI     ' '     ;CHECK FOR AN ERROR
06D1 C8                RZ              ;RETURN IF NO ERROR
06D2 21B210    AOU1:   LXI     H,OBUF  ;OUTPUT BUFFER ADDRESS
06D5 CD7A02            CALL    SCRN    ;OUTPUT LINE...
06D8 CD0E01            CALL    CRLF
06DB C9                RET
                       ;
                       ; PASS1 OF ASSEMBLER. USED TO FORM SYMBOL TABLE
                       ;
06DC CD6601    PAS1:   CALL    ZBUF    ;CLEAR BUFFER
06DF 329410            STA     PASI    ;SET FOR PASS1
06E2 21C710            LXI     H,IBUF  ;INITIALIZE LINE POINTER
06E5 229610            SHLD    PNTR
06E8 7E                MOV     A,M     ;FETCH CHARACTER
06E9 FE20              CPI     ' '     ;CHECK FOR A BLANK
06EB CA1E07            JZ      OPC     ;JUMP IF NO LABEL
06EE FE2A              CPI     '*'     ;CHECK FOR COMMENT
06F0 C8                RZ              ;RETURN IF COMMENT
                       ;
                       ; PROCESS LABEL
                       ;
06F1 CD200B            CALL    SLAB    ;GET AND CHECK LABEL
06F4 DADF0A            JC      OP5     ;ERROR IN LABEL
06F7 CAC70C            JZ      ERRD    ;DUPLICATE LABEL
06FA CD3507            CALL    LCHK    ;CHECK CHARACTER AFTER LABEL
06FD C2DF0A            JNZ     OP5     ;ERROR IF NO BLANK
0700 0E05              MVI     C,LLAB  ;LENGTH OF LABELS
0702 217E10            LXI     H,ABUF  ;SET BUFFER ADDRESS
0705 7E        MLAB:   MOV     A,M     ;FETCH NEXT CHARACTER
0706 12                STAX    D       ;STORE IN SYMBOL TABLE
0707 13                INX     D
0708 23                INX     H
0709 0D                DCR     C
070A C20507            JNZ     MLAB
070D EB                XCHG
070E 229010            SHLD    TABA    ;SAVE TABLE ADDRESS FOR EQU
0711 3A9310            LDA     ASPC+1  ;FETCH PC (HIGH)
0714 77                MOV     M,A
0715 23                INX     H
0716 3A9210            LDA     ASPC    ;FETCH PC (LOW)
0719 77                MOV     M,A     ;STORE IN TABLE
071A 219810            LXI     H,NOLA
071D 34                INR     M       ;INCREMENT NUMBER OF LABELS
                       ;
                       ; PROCESS OPCODE
                       ;
071E CD6601    OPC:    CALL    ZBUF    ;ZERO WORKING BUFFER
0721 CD0D09            CALL    SBLK    ;SCAN TO OPCODE
0724 DA060B            JC      OERR    ;FOUND CARRIAGE RETURN
0727 CD750B            CALL    ALPS    ;PLACE OPCODE IN BUFFER
072A FE20              CPI     ' '     ;CHECK FOR BLANK AFTER OPCODE
072C DA650A            JC      OPCD    :CR AFTER OPCODE
```

```
072F C2060B             JNZ     OERR    ;ERROR IF NO BLANK
0732 C3650A             JMP     OPCD    ;CHECK OPCODE
                ;
                ; THIS ROUTINE CHECKS THE CHARACTER AFTER A LABEL
                ; FOR A BLANK OR A COLON.
                ;
0735 2A9610     LCHK:   LHLD    PNTR
0738 7E                 MOV     A,M     ;GET CHARACTER AFTER LABEL
0739 FE20               CPI     ' '     ;CHECK FOR A BLANK
073B C8                 RZ              ;RETURN IF A BLANK
073C FE3A               CPI     ':'     ;CHECK FOR A COLON
073E C0                 RNZ
073F 23                 INX     H
0740 229610             SHLD    PNTR    ;SAVE POINTER
0743 C9                 RET
                ;
                ; PROCESS ANY PSEUDO OPS THAT NEED TO BE IN PASS 1
                ;
0744 CD0D09     PSU1:   CALL    SBLK    ;SCAN TO OPERAND
0747 1A                 LDAX    D       ;FETCH VALUE
0748 B7                 ORA     A       ;SET FLAGS
0749 CA6007             JZ      ORG1    ;ORG OPCODE
074C FA9007             JM      DAT1    ;DATA STATEMENT
074F E27507             JPO     EQU1    ;EQU OPCODE
0752 FE05               CPI     5
0754 DA8807             JC      RES1    ;RES OPCODE
0757 C20109             JNZ     EASS    ;JUMP IF END
                ; DO DW PSEUDO/OP
075A 0E02       ACO1:   MVI     C,2     ;2 BYTE INSTRUCTION
075C AF                 XRA     A       ;GET A ZERO
075D C3F50A             JMP     OCN1    ;ADD VALUE TO PROGRAM CNTR
                ; DO ORG PSEUDO-OP
0760 CD970B     ORG1:   CALL    ASCN    ;GET OPERAND
0763 3AB210             LDA     OBUF    ;FETCH ERROR INDICATOR
0766 FE20               CPI     ' '     ;CHECK FOR AN ERROR
0768 C0                 RNZ
0769 229210             SHLD    ASPC    ;STORE NEW ORIGIN
076C 3AC710             LDA     IBUF    ;GET FIRST CHARACTER
076F FE20               CPI     ' '     ;CHECK FOR LABEL
0771 C8                 RZ              ;NO LABEL
0772 C38007             JMP     EQUS    ;CHANGE LABEL VALUE
                ; DO EQU PSEUDO-OP
0775 CD970B     EQU1:   CALL    ASCN    ;GET OPERAND
0778 3AC710             LDA     IBUF    ;FETCH 1ST CHARACTER
077B FE20               CPI     ' '     ;CHECK FOR LABEL
077D CA9F0C             JZ      ERRM    ;MISSING LABEL
0780 EB         EQUS:   XCHG
0781 2A9010             LHLD    TABA    ;SYMBOL TABLE ADDRESS
0784 72                 MOV     M,D     ;STORE LABEL VALUE
0785 23                 INX     H
0786 73                 MOV     M,E
0787 C9                 RET
                ; DO DS PSEUDO-OP
0788 CD970B     RES1:   CALL    ASCN    ;GET OPERAND
078B 44                 MOV     B,H
078C 4D                 MOV     C,L
078D C3ED07             JMP     RES21   ;ADD VALUE TO PROGRAM COUNTER
                ;
                ; DO DB PSEUDO-OP
                ;
0790 C3F407     DAT1:   JMP     DAT2A
                ;
                ; PERFORM PASS 2 OF THE ASSEMBLER
                ;
0793 21B410     PAS2:   LXI     H,OBUF+2 ;SET OUTPUT BUFFER ADDRESS
0796 3A9310             LDA     ASPC+1  ;FETCH PC(HIGH)
```

```
0799 CD8902            CALL    BINH+3  ;CONVERT FOR OUTPUT
079C 23                INX     H
079D 3A9210            LDA     ASPC    ;FETCH PC(LOW)
07A0 CD8902            CALL    BINH+3  ;CONVERT FOR OUTPUT
07A3 23                INX     H
07A4 229E10            SHLD    OIND    ;SAVE OUTPUT ADDRESS
07A7 CD6601            CALL    ZBUF    ;CLEAR BUFFER
07AA 21C710            LXI     H,IBUF  ;INITIALIZE LINE POINTER
07AD 229610    PABL:   SHLD    PNTR    ;SAVE POINTER
07B0 7E                MOV     A,M     ;FETCH FIRST CHARACTER
07B1 FE20              CPI     ' '     ;CHECK FOR LABEL
07B3 CA1E07            JZ      OPC     ;GET OPCODE
07B6 FE2A              CPI     '*'     ;CHECK FOR COMMENT
07B8 C8                RZ              ;RETURN IF COMMENT
07B9 CD200B            CALL    SLAB    ;SCAN OFF LABEL
07BC DAC20C            JC      ERRL    ;ERROR IN LABEL
07BF CD3507            CALL    LCHK    ;CHECK FOR A BLANK OR COLON
07C2 C2C20C            JNZ     ERRL    ;ERROR IF NOT A BLANK
07C5 C31E07            JMP     OPC
                ;
                ;
                ;
                ; PROCESS PSEUDO OPS FOR PASS2
07C8 1A        PSU2:   LDAX    D
07C9 B7                ORA     A       ;SET FLAGS
07CA CA0C08            JZ      ORG2    ;ORG OPCODE
07CD FAF107            JM      DAT2    ;DATA OPCODE
07D0 E2FA07            JPO     EQU2    ;EQUATE PSEUDE-OP
07D3 FE05              CPI     5
07D5 DAE107            JC      RES2    ;RES OPCODE
07D8 C20109            JNZ     EASS    ;END OPCODE
                ; DO DW PSEUDO-OP
07DB CDE108    ACO2:   CALL    TYS6    ;GET VALUE
07DE C35A07            JMP     ACO1
                ; DO DS PSEUDO-OP
07E1 CD940B    RES2:   CALL    ASBL    ;GET OPERAND
07E4 44                MOV     B,H
07E5 4D                MOV     C,L
07E6 2A8C10            LHLD    BBUF+2  ;FETCH STORAGE COUNTER
07E9 09                DAD     B       ;ADD VALUE
07EA 228C10            SHLD    BBUF+2
07ED AF        RES21:  XRA     A       ;GET A ZERO
07EE C3F80A            JMP     OCN2
                ; DO DB PSEUDO-OP
07F1 CDA008    DAT2:   CALL    TYS5    ;GET OPERAND
07F4 AF        DAT2A:  XRA     A       ;MAKE A ZERO
07F5 0E01              MVI     C,1     ;BYTE COUNT
07F7 C3F50A            JMP     OCN1
                ;
                ; HANDLE EQUATES ON 2ND PASS.
                ;
07FA CD940B    EQU2:   CALL    ASBL    ;GET OPERAND INTO HL AND
                                        ;  FALL INTO NEXT ROUTINE
                ;
                ; STORE CONTENTS OF HL AS HEX ASCII AT OBUF+2.
                ;   ON RETURN, DE HOLDS VALUE WHICH WAS IN HL.
                ;
07FD EB        BINAD:  XCHG            ;PUT VALUE INTO DE
07FE 21B410            LXI     H,OBUF+2 ;POINTER TO ADDR IN OBUF
0801 7A                MOV     A,D     ;STORE HI BYTE....
0802 CD8902            CALL    BINH+3
0805 23                INX     H
0806 7B                MOV     A,E     ;STORE LO BYTE...
0807 CD8902            CALL    BINH+3
080A 23                INX     H
080B C9                RET
```

```
                        ; DO ORG PSEUDO-OP
080C CD940B     ORG2:   CALL    ASBL    ;GET NEW ORIGIN
080F 3AB210             LDA     OBUF    ;GET ERROR INDICATOR
0812 FE20               CPI     ' '     ;CHECK FOR AN ERROR
0814 C0                 RNZ             ;DON'T MODIFY PC IF ERROR
0815 CDFD07             CALL    BINAD   ;STORE NEW ADDR IN OBUF
0818 2A9210             LHLD    ASPC    ;FETCH PC
081B EB                 XCHG
081C 229210             SHLD    ASPC    ;STORE NEW PC
081F 7D                 MOV     A,L
0820 93                 SUB     E       ;FORM DIFFERENCE OF ORIGINS
0821 5F                 MOV     E,A
0822 7C                 MOV     A,H
0823 9A                 SBB     D
0824 57                 MOV     D,A
0825 2A8C10             LHLD    BBUF+2  ;FETCH STORAGE POINTER
0828 19                 DAD     D       ;MODIFY
0829 228C10             SHLD    BBUF+2  ;SAVE
082C C9                 RET
                        ;
                        ; PROCESS 1 BYTE INSTRUCTIONS WITHOUT OPERANDS
                        ;
082D CDEE08     TYP1:   CALL    ASTO    ;STORE VALUE IN MEMORY
0830 C9                 RET
                        ;
                        ; PROCESS STAX AND LDAX INSTRUCTIONS
                        ;
0831 CD940B     TYP2:   CALL    ASBL    ;FETCH OPERAND
0834 C4810C             CNZ     ERRR    ;ILLEGAL REGISTER
0837 7D                 MOV     A,L     ;GET LOW ORDER OPERAND
0838 B7                 ORA     A       ;SET FLAGS
0839 CA5508             JZ      TY31    ;OPERAND = 0
083C FE02               CPI     2       ;OPERAND = 2
083E C4810C             CNZ     ERRR    ;ILLEGAL REGISTER
0841 C35508             JMP     TY31
                        ;
                        ; PROCESS PUSH,POP,INX,DCX,DAD INSTRUCTIONS
                        ;
0844 CD940B     TYP3:   CALL    ASBL    ;FETCH OPERAND
0847 C4810C             CNZ     ERRR    ;ILLEGAL REGISTER
084A 7D                 MOV     A,L     ;GET LOW ORDER OPERAND
084B 0F                 RRC             ;CHECK LOW ORDER BIT
084C DC810C             CC      ERRR    ;ILLEGAL REGISTER
084F 17                 RAL             ;RESTORE
0850 FE08               CPI     8
0852 D4810C             CNC     ERRR    ;ILLEGAL REGISTER
0855 07         TY31:   RLC             ;MULTIPLY BY 8
0856 17                 RAL
0857 17                 RAL
0858 47         TY32:   MOV     B,A
0859 1A                 LDAX    D       ;FETCH OPCODE BASE
085A 80                 ADD     B       ;FORM OPCODE
085B FE76               CPI     118     ;CHECK FOR MOV M,M
085D CC810C             CZ      ERRR    ;ILLEGAL REGISTER
0860 C32D08             JMP     TYP1
                        ;
                        ; PROCESS ACCUMULATOR, INR,DCR,MOV,RST INSTRUCTIONS
                        ;
0863 CD940B     TYP4:   CALL    ASBL    ;FETCH OPERAND
0866 C4810C             CNZ     ERRR    ;ILLEGAL REGISTER
0869 7D                 MOV     A,L     ;GET LOW ORDER OPERAND
086A FE08               CPI     8
086C D4810C             CNC     ERRR    ;ILLEGAL REGISTER
086F 1A                 LDAX    D       ;FETCH OPCODE BASE
0870 FE40               CPI     64      ;CHECK FOR MOV INSTRUCTION
0872 CA8108             JZ      TY41
```

```
0875 FEC7              CPI     199
0877 7D                MOV     A,L
0878 CA5508            JZ      TY31    ;RST INSRUCTION
087B FA5808            JM      TY32    ;ACCUMULATOR INSTRUCTION
087E C35508            JMP     TY31    ;INR,DCR
                ; PROCESS MOV INSTRUCTION
0881 29         TY41:  DAD     H       ;MULTIPLY OPERAND BY 8
0882 29                DAD     H
0883 29                DAD     H
0884 85                ADD     L       ;FORM OPCODE
0885 12                STAX    D       ;SAVE OPCODE
0886 CDBF08            CALL    MPNT
0889 CD970B            CALL    ASCN
088C C4810C            CNZ     ERRR    ;INCREMENT POINTER
088F 7D                MOV     A,L     ;FETCH LOW ORDER OPERAND
0890 FE08              CPI     8
0892 D4810C            CNC     ERRR    ;ILLEGAL REGISTER
0895 C35808            JMP     TY32
                ;
                ; PROCESS IMMEDIATE INSTRUCTIONS
                ; IMMEDIATE BYTE CAN BETWEEN -256 AND +255
                ; MVI INSTRUCTION IS A SPECIAL CASE AND CONTAINS
                ; 2 ARGUMENTS IN OPERAND
0898 FE06       TYP5:  CPI     6       ;CHECK FOR MVI INSTRUCTION
089A CCAD08            CZ      TY56
089D CDEE08            CALL    ASTO    ;STORE OBJECT BYTE
08A0 CD940B     TY55:  CALL    ASBL    ;GET IMMEDIATE ARGUMENT
08A3 3C                INR     A
08A4 FE02              CPI     2       ;CHECK OPERAND FOR RANGE
08A6 D49A0C            CNC     ERRV    ;OPERAND OUT OF RANGE
08A9 7D                MOV     A,L
08AA C32D08            JMP     TYP1
                ;
                ; FETCH 1ST ARG FOR MVI AND LXI INSTRUCTIONS
                ;
08AD CD940B     TY56:  CALL    ASBL    ;FETCH ARG
08B0 C4810C            CNZ     ERRR    ;ILLEGAL REGISTER
08B3 7D                MOV     A,L     ;GET LOW ORDER ARGUMENT
08B4 FE08              CPI     8
08B6 D4810C            CNC     ERRR    ;ILLEGAL REGISTER
08B9 29                DAD     H
08BA 29                DAD     H
08BB 29                DAD     H
08BC 1A                LDAX    D       ;FETCH OPCODE BASE
08BD 85                ADD     L       ;FOR OPCODE
08BE 5F                MOV     E,A     ;SAVE OBJECT BYTE
08BF 2A9610     MPNT:  LHLD    PNTR    ;FETCH POINTER
08C2 7E                MOV     A,M     ;FETCH CHARACTER
08C3 FE2C              CPI     ','     ;CHECK FOR COMMA
08C5 23                INX     H       ;INCREMENT POINTER
08C6 229610            SHLD    PNTR
08C9 C28A0C            JNZ     ERRS    ;SYNTAX ERROR IF NO COMMA
08CC 7B                MOV     A,E
08CD C9                RET
                ;
                ; PROCESS 3 BYTE INSTRUCTIONS
                ; LXI INSTRUCTION IS A SPECIAL CASE
                ;
08CE FE01       TYP6:  CPI     1       ;CHECK FOR LXI INSTRUCTION
08D0 C2DE08            JNZ     TY6     ;JUMP IF NOT LXI
08D3 CDAD08            CALL    TY56    ;GET REGISTER
08D6 E608              ANI     08H     ;CHECK FOR ILLEGAL REGISTER
08D8 C4810C            CNZ     ERRR    ;REGISTER ERROR
08DB 7B                MOV     A,E     ;GET OPCODE
08DC E6F7              ANI     0F7H    ;CLEAR BIT IN ERROR
08DE CDEE08     TY6:   CALL    ASTO    :STORE OBJECT BYTE
```

```
08E1 CD940B   TYS6:   CALL    ASBL    ;FETCH OPERAND
08E4 7D               MOV     A,L
08E5 54               MOV     D,H
08E6 CDEE08           CALL    ASTO    ;STORE 2ND BYTE
08E9 7A               MOV     A,D
08EA C32D08           JMP     TYP1
08ED C9               RET
                      ;
                      ; THIS ROUTINE IS USED TO STORE OBJECT CODE PRODUCED
                      ; BY THE ASSEMBLER DURING PASS 2 INTO MEMORY
                      ;
08EE 2A8C10   ASTO:   LHLD    BBUF+2  ;FETCH STORAGE ADDRESS
08F1 77               MOV     M,A     ;STORE OBJECT BYTE
08F2 23               INX     H       ;INCREMENT LOCATION
08F3 228C10           SHLD    BBUF+2
08F6 2A9E10           LHLD    OIND    ;FETCH OUTPUT ADDRESS
08F9 23               INX     H
08FA CD8902           CALL    BINH+3  ;CONVERT OBJECT BYTE
08FD 229E10           SHLD    OIND
0900 C9               RET
                      ;
                      ; GET HERE WHEN END PSEUDO-OP IS FOUND OR WHEN
                      ; END-OF-FILE OCCURS IN SOURCE FILE. CONTROL IS SET
                      ; FOR EITHER PASS 2 OR ASSEMBLY TERMINATOR IF FINISHED.
                      ;
0901 3A9410   EASS:   LDA     PASI    ;FETCH PASS INDICATOR
0904 B7               ORA     A       ;SET FLAGS
0905 C26700           JNZ     EOR     ;JUMP IF FINISHED
0908 3E01             MVI     A,1     ;PASS INDICATOR FOR 2ND PASS
090A C37A06           JMP     ASM3    ;DO 2ND PASS
                      ;
                      ; THIS ROUTINE SCANS THROUGH A CHARACTER STRING UNTIL
                      ; THE FIRST NON-BLANK CHARACTER IS FOUND
                      ;
                      ; ON RETURN, CARRY SET INDICATES A CARRIAGE RETURN
                      ; AS FIRST NON-BLANK CHARACTER.
                      ;
090D 2A9610   SBLK:   LHLD    PNTR    ;FETCH ADDRESS
0910 7E       SBL1:   MOV     A,M     ;FETCH CHARACTER
0911 FE20             CPI     ' '     ;CHECK FOR A BLANK
0913 C0               RNZ             ;RETURN IF NON-BLANK
0914 23       SBL2:   INX     H       ;INCREMENT
0915 229610           SHLD    PNTR    ;SAVE POINTER
0918 C31009           JMP     SBL1
                      ;
                      ;
                      ; THIS ROUTINE IS USED TO CHECK THE CONDITION
                      ; CODE MNEMONICS FOR CONDITIONAL JUMPS, CALLS,
                      ; AND RETURNS.
                      ;
091B 217F10   COND:   LXI     H,ABUF+1
091E 227410           SHLD    ADDS
0921 0602             MVI     B,2     ;2 CHARACTERS
0923 CD500A           CALL    COPC
0926 C9               RET
                      ;
                      ;
                      ; THE FOLLOWING IS THE OPCODE TABLE
                      ;
                      ;
0927 4F5247   OTAB:   DB      'ORG'
092A 00               DB      0
092B 00               DB      0
092C 455155           DB      'EQU'
092F 00               DB      0
0930 01               DB      1
```

```
0931 4442          DB      'DB'
0933 00            DB      0
0934 00            DB      0
0935 FF            DB      -1
0936 4453          DB      'DS'
0938 00            DB      0
0939 00            DB      0
093A 03            DB      3
093B 4457          DB      'DW'
093D 00            DB      0
093E 00            DB      0
093F 05            DB      5
0940 454E44        DB      'END'
0943 00            DB      0
0944 06            DB      6
0945 00            DB      0
0946 484C54        DB      'HLT'
0949 76            DB      118
094A 524C43        DB      'RLC'
094D 07            DB      7
094E 525243        DB      'RRC'
0951 0F            DB      15
0952 52414C        DB      'RAL'
0955 17            DB      23
0956 524152        DB      'RAR'
0959 1F            DB      31
095A 524554        DB      'RET'
095D C9            DB      201
095E 434D41        DB      'CMA'
0961 2F            DB      47
0962 535443        DB      'STC'
0965 37            DB      55
0966 444141        DB      'DAA'
0969 27            DB      39
096A 434D43        DB      'CMC'
096D 3F            DB      63
096E 4549          DB      'EI'
0970 00            DB      0
0971 FB            DB      251
0972 4449          DB      'DI'
0974 00            DB      0
0975 F3            DB      243
0976 4E4F50        DB      'NOP'
0979 00            DB      0
097A 00            DB      0
097B 58434847      DB      'XCHG'
097F EB            DB      235
0980 5854484C      DB      'XTHL'
0984 E3            DB      227
0985 5350484C      DB      'SPHL'
0989 F9            DB      249
098A 5043484C      DB      'PCHL'
098E E9            DB      233
098F 00            DB      0
0990 53544158      DB      'STAX'
0994 02            DB      2
0995 4C444158      DB      'LDAX'
0999 0A            DB      10
099A 00            DB      0
099B 50555348      DB      'PUSH'
099F C5            DB      197
09A0 504F50        DB      'POP'
09A3 00            DB      0
09A4 C1            DB      193
09A5 494E58        DB      'INX'
09A8 00            DB      0
```

```
09A9 03              DB      3
09AA 444358          DB      'DCX'
09AD 00              DB      0
09AE 0B              DB      11
09AF 444144          DB      'DAD'
09B2 00              DB      0
09B3 09              DB      9
09B4 00              DB      0
09B5 494E52          DB      'INR'
09B8 04              DB      4
09B9 444352          DB      'DCR'
09BC 05              DB      5
09BD 4D4F56          DB      'MOV'
09C0 40              DB      64
09C1 414444          DB      'ADD'
09C4 80              DB      128
09C5 414443          DB      'ADC'
09C8 88              DB      136
09C9 535542          DB      'SUB'
09CC 90              DB      144
09CD 534242          DB      'SBB'
09D0 98              DB      152
09D1 414E41          DB      'ANA'
09D4 A0              DB      160
09D5 585241          DB      'XRA'
09D8 A8              DB      168
09D9 4F5241          DB      'ORA'
09DC B0              DB      176
09DD 434D50          DB      'CMP'
09E0 B8              DB      184
09E1 525354          DB      'RST'
09E4 C7              DB      199
09E5 00              DB      0
09E6 414449          DB      'ADI'
09E9 C6              DB      198
09EA 414349          DB      'ACI'
09ED CE              DB      206
09EE 535549          DB      'SUI'
09F1 D6              DB      214
09F2 534249          DB      'SBI'
09F5 DE              DB      222
09F6 414E49          DB      'ANI'
09F9 E6              DB      230
09FA 585249          DB      'XRI'
09FD EE              DB      238
09FE 4F5249          DB      'ORI'
0A01 F6              DB      246
0A02 435049          DB      'CPI'
0A05 FE              DB      254
0A06 494E            DB      'IN'
0A08 00              DB      0
0A09 DB              DB      219
0A0A 4F5554          DB      'OUT'
0A0D D3              DB      211
0A0E 4D5649          DB      'MVI'
0A11 06              DB      6
0A12 00              DB      0
0A13 4A4D50          DB      'JMP'
0A16 00              DB      0
0A17 C3              DB      195
0A18 43414C4C        DB      'CALL'
0A1C CD              DB      205
0A1D 4C5849          DB      'LXI'
0A20 00              DB      0
0A21 01              DB      1
0A22 4C4441          DB      'LDA'
```

```
0A25 00                 DB      0
0A26 3A                 DB      58
0A27 535441             DB      'STA'
0A2A 00                 DB      0
0A2B 32                 DB      50
0A2C 53484C44           DB      'SHLD'
0A30 22                 DB      34
0A31 4C484C44           DB      'LHLD'
0A35 2A                 DB      42
0A36 00                 DB      0
                ; CONDITION CODE TABLE
0A37 4E5A               DB      'NZ'
0A39 00                 DB      0
0A3A 5A                 DB      'Z'
0A3B 00                 DB      0
0A3C 08                 DB      8
0A3D 4E43               DB      'NC'
0A3F 10                 DB      16
0A40 43                 DB      'C'
0A41 00                 DB      0
0A42 18                 DB      24
0A43 504F               DB      'PO'
0A45 20                 DB      32
0A46 5045               DB      'PE'
0A48 28                 DB      40
0A49 50                 DB      'P'
0A4A 00                 DB      0
0A4B 30                 DB      48
0A4C 4D                 DB      'M'
0A4D 00                 DB      0
0A4E 38                 DB      56
0A4F 00                 DB      0
                ;
                ; THIS ROUTINE IS USED TO CHECK A GIVEN OPCODE
                ; AGAINST THE LEGAL OPCODES IN THE OPCODE TABLE.
                ;
0A50 2A7410     COPC:   LHLD    ADDS
0A53 1A                 LDAX    D       ;FETCH CHARACTER
0A54 B7                 ORA     A       ;SET FLAGS
0A55 CA620A             JZ      COP1    ;JUMP IF TERMINATION CHARACTER
0A58 48                 MOV     C,B
0A59 CD5301             CALL    SEAR
0A5C 1A                 LDAX    D
0A5D C8                 RZ              ;RETURN IF MATCH
0A5E 13                 INX     D       ; NEXT STRING
0A5F C3500A             JMP     COPC    ;CONTINUE SEARCH
0A62 3C         COP1:   INR     A       ;CLEAR ZERO FLAG
0A63 13                 INX     D       ;INCREMENT ADDRESS
0A64 C9                 RET
                ;
                ; THIS ROUTINE CHECKS THE LEGAL OPCODES IN BOTH PASS 1
                ; AND PASS 2.  IN PASS 1 THE PROGRAM COUNTER IS INCRE-
                ; MENTED BY THE CORRECT NUMBER OF BYTES.  AN ADDRESS IS
                ; ALSO SET SO THAT AN INDEXED JUMP CAN BE MADE TO
                ; PROCESS THE OPCODE FOR PASS 2.
                ;
                ;
0A65 217E10     OPCD:   LXI     H,ABUF  ;GET ADDRESS
0A68 227410             SHLD    ADDS
0A6B 112709             LXI     D,OTAB  ;OPCODE TABLE ADDRESS
0A6E 0604               MVI     B,4     ;CHARACTER COUNT
0A70 CD500A             CALL    COPC    ;CHECK OPCODES
0A73 CA0E0B             JZ      PSEU    ;JUMP IF A PSEUDO-OP
0A76 05                 DCR     B       ;3 CHARACTER OPCODES
0A77 CD500A             CALL    COPC
0A7A CA810A             JZ      OP1
```

```
0A7D 04              INR     B       ;4 CHARACTER  OPCODES
0A7E CD500A          CALL    COPC
0A81 212D08  OP1:    LXI     H,TYP1  ;TYPE 1 INSTRUCTIONS
0A84 0E01    OP2:    MVI     C,1     ;1 BYTE INSTRUCTIONS
0A86 CAE10A          JZ      OCNT
             ;
0A89 CD500A  OPC2:   CALL    COPC    ;CHECK FOR STAX,LDAX
0A8C 213108          LXI     H,TYP2
0A8F CA840A          JZ      OP2
0A92 CD500A          CALL    COPC    ;CHECK FOR PUSH,POP,INX
             ;               ;       ;DCX AND DAD
0A95 214408          LXI     H,TYP3
0A98 CA840A          JZ      OP2
0A9B 05              DCR     B       ;3 CHAR OPCODES
0A9C CD500A          CALL    COPC    ;ACCUMULATOR INSTRUCTIONS,
             ;               ;       ;INR,DCR,MOV,RST
0A9F 216308          LXI     H,TYP4
0AA2 CA840A          JZ      OP2
             ;
0AA5 CD500A  OPC3:   CALL    COPC    ;IMMEDIATE INSTRUCTIONS
0AA8 219808          LXI     H,TYP5
0AAB 0E02            MVI     C,2     ;2 BYTE INSTRUCTIONS
0AAD CAE10A          JZ      OCNT
0AB0 04              INR     B       ;4 CHAR OPCODES
0AB1 CD500A          CALL    COPC    ;JMP,CALL,LXI,LDA,STA,
             ;               ;       ;LHLD,SHLD OPCODES
0AB4 CADC0A          JZ      OP4
0AB7 CD1B09          CALL    COND    ;CONDITIONAL INSTRUCTIONS
0ABA C2060B          JNZ     OERR    ;ILLEGAL OPCODE
0ABD C6C0            ADI     192     ;ADD BASE VALUE TO RETURN
0ABF 57              MOV     D,A
0AC0 0603            MVI     B,3     ;3 CHARACTER OPCODES
0AC2 3A7E10          LDA     ABUF    ;FETCH FIRST CHARACTER
0AC5 4F              MOV     C,A     ;SAVE CHARACTER
0AC6 FE52            CPI     'R'     ;CONDITIONAL RETURN
0AC8 7A              MOV     A,D
0AC9 CA810A          JZ      OP1
0ACC 79              MOV     A,C
0ACD 14              INR     D       ;FORM CONDITIONAL JUMP
0ACE 14              INR     D
0ACF FE4A            CPI     'J'     ;CONDITIONAL JUMP
0AD1 CADB0A          JZ      OPAD
0AD4 FE43            CPI     'C'     ;CONDITIONAL CALL
0AD6 C2060B          JNZ     OERR    ;ILLEGAL OPCODE
0AD9 14              INR     D       ;FORM CONDIITIONAL CALL
0ADA 14              INR     D
0ADB 7A      OPAD:   MOV     A,D     ;GET OPCODE
0ADC 21CE08  OP4:    LXI     H,TYP6
0ADF 0E03    OP5:    MVI     C,3     ;3 BYTE INSTRUCTION
0AE1 329D10  OCNT:   STA     TEMP    ;SAVE OPCODE
             ;
             ; CHECK FOR OPCODE ONLY CONTAINING THE CORRECT NUMBER OF
             ; CHARACTERS. THUS ADDQ, SAY, WOULD GIVE AN ERROR
             ;
0AE4 3E7E            MVI     A,ABUF AND 0FFH ;LOAD BUFFER ADDRESS
0AE6 80              ADD     B       ;ADD LENGTH OF OPCODE
0AE7 5F              MOV     E,A
0AE8 3E10            MVI     A,ABUF/256
0AEA CE00            ACI     0       ;GET HIGH ORDER ADDRESS
0AEC 57              MOV     D,A
0AED 1A              LDAX    D       ;FETCH CHARACTER AFTER OPCODE
0AEE B7              ORA     A       ;IT SHOULD BE ZERO
0AEF C2060B          JNZ     OERR    ;OPCODE ERROR
0AF2 3A9410          LDA     PASI    ;FETCH PASS INDICATOR
0AF5 0600    OCN1:   MVI     B,0
0AF7 EB              XCHG
```

```
0AF8 2A9210    OCN2:   LHLD    ASPC    ;FETCH PROGRAM COUNTER
0AFB 09                DAD     B       ;ADD IN BYTE COUNT
0AFC 229210            SHLD    ASPC    ;STORE PC
0AFF B7                ORA     A       ;WHICH PASS?
0B00 C8                RZ             ;RETURN IF PASS 1
0B01 3A9D10            LDA     TEMP    ;FETCH OPCODE
0B04 EB                XCHG
0B05 E9                PCHL
                ;
0B06 21AD0C    OERR:   LXI     H,ERRO  ;SET ERROR ADDRESS
0B09 0E03              MVI     C,3     ;LEAVE 3 BYTES FOR PATCH
0B0B C3F20A            JMP     OCN1-3
                ;
0B0E 218210    PSEU:   LXI     H,ABUF+4        ;SET BUFFER ADDRESS
0B11 7E                MOV     A,M     ;FETCH CHARACTER AFTER OPCODE
0B12 B7                ORA     A       ;SHOULD BE A ZERO
0B13 C2060B            JNZ     OERR
0B16 3A9410            LDA     PASI    ;FETCH PASS INDICATOR
0B19 B7                ORA     A
0B1A CA4407            JZ      PSU1
0B1D C3C807            JMP     PSU2
                ;
                ;
                ; THIS ROUTINE IS USED TO PROCESS LABELS.
                ; IT CHECKS TO SEE IF A LABEL IS IN THE SYMBOL TABLE
                ; OR NOT.  ON RETURN, Z=1 INDICATES A MATCH WAS FOUND
                ; AND H,L CONTAIN THE VALUE ASSOCIATED WITH THE LABEL.
                ; THE REGISTER NAMES A, B, C, D, E, H, L, P, AND S ARE
                ; PRE-DEFINED AND NEED NOT BE ENTERED BY THE USER.
                ; ON RETURN, C=1 INDICATES A LABEL ERROR.
                ;
0B20 FE41      SLA8:   CPI     'A'     ;CHECK FOR LEGAL CHAR
0B22 D8                RC
0B23 FE5B              CPI     'Z'+1   ;CHECK FOR ILLEGAL CHAR
0B25 3F                CMC
0B26 D8                RC              ;RETURN IF ILLEGAL CHAR
0B27 CD750B            CALL    ALPS    ;PLACE SYMBOL IN BUFFER
0B2A 217E10            LXI     H,ABUF  ;SET BUFFER ADDRESS
0B2D 227410            SHLD    ADDS    ;SAVE ADDDRESS
0B30 05                DCR     B       ;CHECK IF ONE CHARACTER
0B31 C2440B            JNZ     SLA1
                ; CHECK IF PREDEFINED REGISTER NAME
0B34 04                INR     B       ;SET B=1
0B35 11600B            LXI     D,RTAB  ;REGISTER TABLE ADDRESS
0B38 CD500A            CALL    COPC    ;CHECK NAME OF REGISTER
0B3B C2440B            JNZ     SLA1    ;NOT A PREDEFINED REGISTER
0B3E 6F                MOV     L,A     ;SET VALUE(HIGH)
0B3F 2600              MVI     H,0
0B41 C35A0B            JMP     SLA2
0B44 3A9810    SLA1:   LDA     NOLA    ;FETCH SYMBOL COUNT
0B47 47                MOV     B,A
0B48 111A11            LXI     D,SYMT  ;SET SYMBOL TABLE ADDRESS
0B4B B7                ORA     A       ;ARE THERE ANY LABELS?
0B4C CA5D0B            JZ      SLA3    ;JUMP IF NO LABELS
0B4F 3E05              MVI     A,LLAB  ;FETCH LENGTH OF LABEL
0B51 329510            STA     NCHR
0B54 CD3C01            CALL    COMS    ;CHECK TABEL
0B57 4C                MOV     C,H     ;SWAP H AND L
0B58 65                MOV     H,L
0B59 69                MOV     L,C
0B5A 37        SLA2:   STC             ;SET CARRY
0B5B 3F                CMC             ;CLEAR CARRY
0B5C C9                RET             ;RETURN
0B5D 3C        SLA3:   INR     A       ;CLEAR ZERO FLAG
0B5E B7                ORA     A       ;CLEAR CARRY
0B5F C9                RET
```

```
                        ;
                        ; PREDEFINE REGISTER VALUES IN THIS TABLE
                        ;
0860 41         RTAB:   DB      'A'
0861 07                 DB      7
0862 42                 DB      'B'
0863 00                 DB      0
0864 43                 DB      'C'
0865 01                 DB      1
0866 44                 DB      'D'
0867 02                 DB      2
0868 45                 DB      'E'
0869 03                 DB      3
086A 48                 DB      'H'
086B 04                 DB      4
086C 4C                 DB      'L'
086D 05                 DB      5
086E 4D                 DB      'M'
086F 06                 DB      6
0870 50                 DB      'P'
0871 06                 DB      6
0872 53                 DB      'S'
0873 06                 DB      6
0874 00                 DB      0       ;END OF TABLE INDICATOR
                        ;
                        ; THIS ROUTINE SCANS THE INPUT LINE AND PLACES THE
                        ; OPCODES AND LABELS IN THE BUFFER.  THE SCAN TERMINATES
                        ; WHEN A CHARACTER OTHER THAN 0-9 OR A-Z IS FOUND.
                        ;
0875 0600       ALPS:   MVI     B,0     ;SET COUNT
0877 12         ALP1:   STAX    D       ;STORE CHARACTER IN BUFFER
0878 04                 INR     B       ;INCREMENT COUNT
0879 78                 MOV     A,B     ;FETCH COUNT
087A FE0B               CPI     11      ;MAXIMUM BUFFER SIZE
087C D0                 RNC             ;RETURN IF BUFFER FILLED
087D 13                 INX     D       ;INCREMENT BUFFER
087E 23                 INX     H       ;INCREMENT INPUT POINTER
087F 229610             SHLD    PNTR    ;SAVE LINE POINTER
0882 7E                 MOV     A,M     ;FETCH CHARACTER
0883 FE30               CPI     '0'     ;CHECK FOR LEGAL CHARACTERS
0885 D8                 RC
0886 FE3A               CPI     '9'+1
0888 DA7708             JC      ALP1
088B FE41               CPI     'A'
088D D8                 RC
088E FE5B               CPI     'Z'+1
0890 DA7708             JC      ALP1
0893 C9                 RET
                        ; THIS ROUTINE IS USED TO SCAN THROUGH THE INPUT LINE
                        ; TO FETCH THE VALUE OF THE OPERAND FIELD.  ON RETURN,
                        ; THE VALUE OF THE OPERAND IS CONTAINED IN REG'S H,L.
                        ;
0894 CD0D09     ASBL:   CALL    SBLK    ;GET FIRST ARGUMENT
0897 210000     ASCN:   LXI     H,0     ;GET A ZERO
089A 229A10             SHLD    OPRD    ;INITIALIZE OPERAND
089D 24                 INR     H
089E 229B10             SHLD    OPRI-1  ;INITIALIZE OPERAND INDICATOR
08A1 2A9610     NXT1:   LHLD    PNTR    ;FETCH SCAN POINTER
08A4 2B                 DCX     H
08A5 CD6601             CALL    ZBUF    ;CLEAR BUFFER
08A8 329910             STA     SIGN    ;ZERO SIGN INDICATOR
08AB 23         NXT2:   INX     H       ;INCREMENT POINTER
08AC 7E                 MOV     A,M     ;FETCH NEXT CHARACTER
08AD FE21               CPI     ' '+1
08AF DA530C             JC      SEND    ;JUMP IF CR OR BLANK
08B2 FE2C               CPI     ','     ;FIELD SEPARATOR
```

```
0BB4 CA530C             JZ      SEND
                ; CHECK FOR OPERATORS
0BB7 FE2B               CPI     '+'     ;CHECK FOR PLUS
0BB9 CAC40B             JZ      ASC1
0BBC FE2D               CPI     '-'     ;CHECK FOR MINUS
0BBE C2D40B             JNZ     ASC2
0BC1 329910             STA     SIGN
0BC4 3A9C10     ASC1:   LDA     OPRI    ;FETCH OPERAND INDICATOR
0BC7 FE02               CPI     2       ;CHECK FOR TWO OPERATORS
0BC9 CA8A0C             JZ      ERRS    ;SYNTAX ERROR
0BCC 3E02               MVI     A,2
0BCE 329C10             STA     OPRI    ;SET INDICATOR
0BD1 C3AB0B             JMP     NXT2
                ;CHECK FOR OPERANDS
0BD4 4F         ASC2:   MOV     C,A     ;SAVE CHARACTER
0BD5 3A9C10             LDA     OPRI    ;GET INDICATOR
0BD8 B7                 ORA     A       ;CHECK FOR TWO OPERANDS
0BD9 CA8A0C             JZ      ERRS    ;SYNTAX ERROR
0BDC 79                 MOV     A,C
0BDD FE24               CPI     '$'     ;LC EXPRESSION
0BDF C2EC0B             JNZ     ASC3
0BE2 23                 INX     H       ;INCREMENT POINTER
0BE3 229610             SHLD    PNTR    ;SAVE POINTER
0BE6 2A9210             LHLD    ASPC    ;FETCH LOCATION COUNTER
0BE9 C3280C             JMP     AVAL
                ;CHECK FOR ASCII CHARACTERS
0BEC FE27       ASC3:   CPI     27H     ;CHECK FOR SINGLE QUOTE
0BEE C2180C             JNZ     ASC5    ;JUMP IF NOT QUOTE
0BF1 110000             LXI     D,0     ;GET A ZERO
0BF4 0E03               MVI     C,3     ;CHARACTER COUNT
0BF6 23         ASC4:   INX     H       ;BUMP POINTER
0BF7 229610             SHLD    PNTR    ;SAVE
0BFA 7E                 MOV     A,M     ;FETCH NEXT CHARACTER
0BFB FE0D               CPI     ASCR    ;IS IT A CR?
0BFD CAA80C             JZ      ERRA    ;ARGUMENT ERROR
0C00 FE27               CPI     27H     ;IS IT QUOTE
0C02 C20F0C             JNZ     SSTR
0C05 23                 INX     H       ;INCREMENT POINTER
0C06 229610             SHLD    PNTR    ;SAVE
0C09 7E                 MOV     A,M     ;FETCH NEXT CHAR
0C0A FE27               CPI     27H     ;CHECK FOR 2 QUOTES IN A ROW
0C0C C2290C             JNZ     AVAL+1  ;TERMINAL QUOTE
0C0F 0D         SSTR:   DCR     C       ;CHECK COUNT
0C10 CAA80C             JZ      ERRA    ;TOO MANY CHARACTERS
0C13 53                 MOV     D,E
0C14 5F                 MOV     E,A     ;SET CHARACTER IN BUFFER
0C15 C3F60B             JMP     ASC4
0C18 FE30       ASC5:   CPI     '0'     ;CHECK FOR NUMERIC
0C1A DAA80C             JC      ERRA    ;ILLEGAL CHARACTER
0C1D FE3A               CPI     '9'+1
0C1F D2470C             JNC     ALAB
0C22 CD630C             CALL    NUMS    ;GET NUMERIC VALUE
0C25 DAA80C             JC      ERRA    ;ARGUMENT ERROR
0C28 EB         AVAL:   XCHG
0C29 2A9A10             LHLD    OPRD    ;FETCH OPERAND
0C2C AF                 XRA     A       ;GET A ZERO
0C2D 329C10             STA     OPRI    ;STOR IN OPERAND INDICATOR
0C30 3A9910             LDA     SIGN    ;GET SIGN INDICATOR
0C33 B7                 ORA     A       ;SET FLAGS
0C34 C23E0C             JNZ     ASUB
0C37 19                 DAD     D       ;FORM RESULT
0C38 229A10     ASC7:   SHLD    OPRD    ;SAVE RESULT
0C3B C3A10B             JMP     NXT1
0C3E 7D         ASUB:   MOV     A,L
0C3F 93                 SUB     E
0C40 6F                 MOV     L,A
```

```
0C41 7C              MOV     A,H
0C42 9A              SBB     D
0C43 67              MOV     H,A
0C44 C3380C          JMP     ASC7
0C47 CD200B  ALAB:   CALL    SLAB
0C4A CA280C          JZ      AVAL
0C4D DAA80C          JC      ERRA    ;ILLEGAL SYMBOL
0C50 C3950C          JMP     ERRU    ;UNDEFINED SYMBOL
                     ;
                     ; GET HERE WHEN TERMINATING CHARACTER IS FOUND.
                     ; CHECK FOR LEADING FIELD SEPARATOR.
                     ;
0C53 3A9C10  SEND:   LDA     OPRI    ;FETCH OPERAND INDICATOR
0C56 B7              ORA     A       ;SET FLAGS
0C57 C28A0C          JNZ     ERRS    ;SYNTAX ERROR
0C5A 2A9A10          LHLD    OPRD
0C5D 7C      SEN1:   MOV     A,H     ;GET HIGH ORDER BYTE
0C5E 119D10          LXI     D,TEMP  ;GET ADDRESS
0C61 B7              ORA     A       ;SET FLAGS
0C62 C9              RET
                     ;
                     ; GET A NUMERIC VALUE WHICH IS EITHER HEXADECIMAL OR
                     ; DECIMAL.  ON RETURN, CARRY SET INDICATES AN ERROR.
                     ;
0C63 CD750B  NUMS:   CALL    ALPS    ;GET NUMERIC
0C66 1B              DCX     D
0C67 1A              LDAX    D       ;GET LAST CHARACTER
0C68 017E10          LXI     B,ABUF  ;SET BUFFER ADDRESS
0C6B FE48            CPI     'H'     ;IS IT HEXADECIMAL?
0C6D CA7B0C          JZ      NUM2
0C70 FE44            CPI     'D'     ;IS IT DECIMAL
0C72 C2770C          JNZ     NUM1
0C75 AF              XRA     A       ;GET A ZERO
0C76 12              STAX    D       ;CLEAR D FROM BUFFER
0C77 CD0102  NUM1:   CALL    ADEC    ;CONVERT DECIMAL VALUE
0C7A C9              RET
0C7B AF      NUM2:   XRA     A       ;GET A ZERO
0C7C 12              STAX    D       ;CLEAR H FROM BUFFER
0C7D CD1B02          CALL    AHEX
0C80 C9              RET
                     ;
                     ; PROCESS REGISTER ERROR
0C81 3E52    ERRR:   MVI     A,'R'   ;GET INDICATOR
0C83 210000          LXI     H,0     ;GET A 0
0C86 32B210          STA     OBUF    ;SET IN OUTPUT BUFFER
0C89 C9              RET
                     ;PROCESS SYNTAX ERROR
0C8A 3E53    ERRS:   MVI     A,'S'   ;GET INDICATOR
0C8C 32B210          STA     OBUF    ;STORE IN OUTPUT BUFFER
0C8F 210000          LXI     H,0
0C92 C35D0C          JMP     SEN1
                     ;PROCESS UNDEFINED SYMBOL ERROR
0C95 3E55    ERRU:   MVI     A,'U'   ;GET INDICATOR
0C97 C38C0C          JMP     ERRS+2
                     ;PROCESS VALUE ERROR
0C9A 3E56    ERRV:   MVI     A,'V'   ;GET INDICATOR
0C9C C3830C          JMP     ERRR+2
                     ;PROCESS MISSING LABEL ERROR
0C9F 3E4D    ERRM:   MVI     A,'M'   ;GET INDICATOR
0CA1 32B210          STA     OBUF    ;STORE IN OUTPUT BUFFER
0CA4 CDD206          CALL    AOU1    ;DISPLAY ERROR
0CA7 C9              RET
                     ;PROCESS ARGUMENT ERROR
0CA8 3E41    ERRA:   MVI     A,'A'   ;GET INDICATOR
0CAA C38C0C          JMP     ERRS+2
                     ; PROCESS OPCODE ERROR
```

```
                        ; STORE 3 BYTES OF ZERO IN OBJECT CODE TO PROVIDE
                        ; FOR A PATCH.
0CAD 3E4F       ERRO:   MVI     A,'O'   ;GET INDICATOR
0CAF 32B210             STA     OBUF    ;STORE IN OUTPUT BUFFER
0CB2 3A9410             LDA     PASI    ;FETCH PASS INDICATOR
0CB5 B7                 ORA     A       ;WHICH PASS
0CB6 C8                 RZ              ;RETURN IF PASS1
0CB7 0E03               MVI     C,3     ;NEED 3 BYTES
0CB9 AF         ERO1:   XRA     A       ;GET A ZERO
0CBA CDEE08             CALL    ASTO    ;PUT IN LISTING AND MEMORY
0CBD 0D                 DCR     C
0CBE C2B90C             JNZ     ERO1
0CC1 C9                 RET
                        ; PROCESS LABEL ERROR
0CC2 3E4C       ERRL:   MVI     A,'L'   ;GET INDICATOR
0CC4 C3AF0C             JMP     ERRO+2
                        ;PROCESS DUPLICATE LABEL ERROR
0CC7 3E44       ERRD:   MVI     A,'D'   ;GET ERROR INDICATOR
0CC9 32B210             STA     OBUF    ;STORE IN OUTPUT BUFFER
0CCC CDC506             CALL    AOUT    ;DISPLAY ERROR
0CCF C31E07             JMP     OPC     ;PROCESS OPCODE
                        ;
                        ;
                        ; THIS ROUTINE SETS OR CLEARS BREAKPOINTS
                        ;
0CD2 3A7E10     BREAK:  LDA     ABUF    ;CHECK FOR AN ARG
0CD5 B7                 ORA     A
0CD6 CA140D             JZ      CLRB    ;IF NO ARG, GO CLEAR BREAKPOINTS
0CD9 1608               MVI     D,NBR   ;ELSE, GET NUMBER OF BREAKPOINTS
0CDB 210C10             LXI     H,BRT   ;AND ADDR OF TABLE
0CDE 7E         B1:     MOV     A,M     ;GET HI BYTE OF ENTRY
0CDF 23                 INX     H
0CE0 46                 MOV     B,M     ;GET LO BYTE OF ENTRY
0CE1 B0                 ORA     B       ;CHECK FOR EMPTY ENTRY
0CE2 CAEE0C             JZ      B2      ;BRANCH IF EMPTY
0CE5 23                 INX     H       ;ELSE GO ON TO NEXT ENTRY
0CE6 23                 INX     H
0CE7 15                 DCR     D       ;BUMP COUNT
0CE8 C2DE0C             JNZ     B1      ;AND TRY AGAIN
0CEB C35A04             JMP     WHAT    ;OOPS! NO ROOM
0CEE 2B         B2:     DCX     H
0CEF EB                 XCHG
0CF0 2A8A10             LHLD    BBUF    ;GET ADDRESS
0CF3 EB                 XCHG            ;IN D,E
0CF4 7A                 MOV     A,D     ;CHECK FOR ADDR > 11D
0CF5 B7                 ORA     A
0CF6 C2FF0C             JNZ     B3
0CF9 7B                 MOV     A,E
0CFA FE0B               CPI     11
0CFC DA5A04             JC      WHAT    ;OOPS. TOO LOW
0CFF 72         B3:     MOV     M,D     ;SAVE ADDRESS
0D00 23                 INX     H
0D01 73                 MOV     M,E
0D02 23                 INX     H
0D03 1A                 LDAX    D       ;PICK UP INSTRUCTION
0D04 77                 MOV     M,A     ;SAVE IT
0D05 3ECF               MVI     A,(RST 1) ;REPLACE IT WITH A
0D07 12                 STAX    D       ;RESTART INSTRUCTION
0D08 3EC3               MVI     A,0C3H  ;SET UP LO MEMORY
0D0A 320800             STA     8       ;WITH A JUMP TO BRKP
0D0D 212E0D             LXI     H,BRKP
0D10 220900             SHLD    9
0D13 C9                 RET             ;THEN RETURN
                        ;
                        ; THIS ROUTINE CLEARS ALL BREAKPOINTS
                        ;
```

```
0D14 210C10   CLRB:   LXI     H,BRT    ;GET TABLE ADDRESS
0D17 0608             MVI     B,NBR    ;GET NUMBER OF BREAKPOINTS
0D19 AF       CLBL:   XRA     A        ;GET A ZERO
0D1A 56               MOV     D,M      ;GET HI-BYTE OF ENTRY
0D1B 77               MOV     M,A
0D1C 23               INX     H
0D1D 5E               MOV     E,M      ;GET LO-BYTE OF ENTRY
0D1E 77               MOV     M,A
0D1F 23               INX     H
0D20 46               MOV     B,M      ;GET INST BYTE
0D21 23               INX     H
0D22 7A               MOV     A,D      ;WAS THIS A NULL ENTRY
0D23 B3               ORA     E
0D24 CA290D           JZ      CL2      ;BRANCH IF IT WAS
0D27 78               MOV     A,B
0D28 12               STAX    D        ;ELSE, PLUG INST BACK IN
0D29 05       CL2:    DCR     B        ;BUMP COUNT
0D2A C2190D           JNZ     CLBL     ;GO DO NEXT ONE
0D2D C9               RET              ;RETURN WHEN DONE
                      ;
                      ; COME HERE WHEN WE HIT A BREAKPOINT
                      ;
0D2E 220810   BRKP:   SHLD    HOLD+8   ;SAVE H,L
0D31 E1               POP     H        ;GET PC
0D32 2B               DCX     H        ;ADJUST IT
0D33 220A10           SHLD    HOLD+10  ;SAVE IT
0D36 F5               PUSH    PSW      ;SAVE FLAGS
0D37 E1               POP     H        ;GET THEM INTO HL
0D38 220010           SHLD    HOLD     ;NOW STORE THEM FOR USER
0D3B 210000           LXI     H,0
0D3E 39               DAD     SP       ;GET STACK POINTER
0D3F 310810           LXI     SP,HOLD+8 ;SET NEW SP
0D42 E5               PUSH    H        ;SAVE OLD SP
0D43 D5               PUSH    D        ;SAVE D,E
0D44 C5               PUSH    B        ;SAVE B,C
0D45 2F               CMA              ;COMPLEMENT ACC
0D46 D3FF             OUT     0FFH     ;DISPLAY IT IN THE LIGHTS
0D48 31B210           LXI     SP,AREA+18 ;SET SP AGAIN
0D4B 2A0A10           LHLD    HOLD+10  ;GET PC
0D4E EB               XCHG             ;INTO D,E
0D4F 210C10           LXI     H,BRT    ;GET ADDR OF TABLE
0D52 0608             MVI     B,NBR    ;AND NUMBER OF ENTRIES
0D54 7E       BL1:    MOV     A,M      ;GET AN ENTRY FROM THE TABLE
0D55 23               INX     H
0D56 BA               CMP     D        ;DOES IT MATCH
0D57 C25F0D           JNZ     BL2      ;BRANCH IF NOT
0D5A 7E               MOV     A,M      ;ELSE GET NEXT BYTE
0D5B BB               CMP     E        ;CHECK IT
0D5C CA680D           JZ      BL3      ;IT MATCHES!
0D5F 23       BL2:    INX     H        ;BUMP AROUND THIS ENTRY
0D60 23               INX     H
0D61 05               DCR     B        ;BUMP COUNT
0D62 CA5A04           JZ      WHAT     ;NOT IN OUR TABLE!
0D65 C3540D           JMP     BL1
                      ;
0D68 23       BL3:    INX     H
0D69 7E               MOV     A,M      ;GET INSTR BYTE
0D6A 12               STAX    D        ;PUT IT BACK
0D6B AF               XRA     A        ;CLEAR ENTRY IN TABLE
0D6C 2B               DCX     H
0D6D 77               MOV     M,A
0D6E 2B               DCX     H
0D6F 77               MOV     M,A
0D70 CD0E01           CALL    CRLF     ;RESTORE THE CARRIAGE
0D73 3A0B10           LDA     HOLD+11  ;GET HI-BYTE OF PC
0D76 CD3A02           CALL    HOUT     ;TYPE IT
```

```
0D79 3A0A10            LDA     HOLD+10 ;GET LO-BYTE OF PC
0D7C CD3A02            CALL    HOUT    ;TYEP IT
0D7F 21880D            LXI     H,BMES  ;TELL USER WHAT IT IS
0D82 CD7A02            CALL    SCRN
0D85 C36700            JMP     EOR     ;GO BACK TO COMMAND LEVEL
                ;
0D88 20425245 BMES:    DB      ' BREAK',13
0D8C 41480D
                ;
                ;
                ; THIS ROUTINE PROCEEDS FROM A BREAKPOINT
                ;
0D8F 3A7E10     PROC:  LDA     ABUF    ;CHECK FOR ARG
0D92 B7                ORA     A
0D93 CA9C0D            JZ      P1      ;JMP IF NO ARG
0D96 2A8A10            LHLD    BBUF    ;ELSE, GET ARG
0D99 220A10            SHLD    HOLD+10 ;PLUG IT INTO PC SLOT
0D9C 310010     P1:    LXI     SP,HOLD ;SET SP TO POINT AT REG'S
0D9F F1                POP     PSW     ;RESTORE PSW
0DA0 C1                POP     B       ;RESTORE B,C
0DA1 D1                POP     D       ;RESTORE D,E
0DA2 E1                POP     H       ;GET OLD SP
0DA3 F9                SPHL            ;RESTORE IT
0DA4 2A0A10            LHLD    HOLD+10 ;GET PC
0DA7 E5                PUSH    H       ;PUT IT ON STACK
0DA8 2A0810            LHLD    HOLD+8  ;RESTORE H,L
0DAB C9                RET             ;AND PROCEED
                ;
                ; SYSTEM RAM
                ;
.0DAC                  ORG     1000H
                ;
                ; DEFINE BREAKPOINT REGION
                ;
0008            NBR    EQU     8       ;NUMBER OF BREAKPOINTS
1000            HOLD:  DS      12      ;REGISTER HOLD AREA
100C            BRT:   DS      3*NBR   ;BREAKPOINT TABLE
                ;
                ; FILE AREA PARAMETERS
0006            MAXFIL EQU     6       ;MAX # OF FILES
0005            NMLEN  EQU     5       ;NAME LENGTH
000D            FELEN  EQU     NMLEN+8 ;DIRECTORY ENTRY LENGTH
1024            FILE0: DS      NMLEN
1029            BOFP:  DS      2
102B            EOFP:  DS      2
102D            MAXL:  DS      4
1031            FILTB: DS      (MAXFIL-1)*FELEN
1072            INSP:  DS      2       ;INSERT LINE POSITION
1072            DELP   EQU     INSP    ;DELETE LINE POSITION
000D            ASCR   EQU     13      ;ASCII CARRIAGE RETURN VALUE
1074            HCON:  DS      2
1074            ADDS   EQU     HCON    ;FIND ADDRESS
1076            FBUF:  DS      NMLEN   ;FILE NAME BUFFER
107B            FREAD: DS      2       ;FREE ADDRESS IN DIRECTORY
107D            FEF:   DS      1       ;FREE ENTRY FOUND FLAG
107D            FOCNT  EQU     FEF     ;OUTPUT COUNTER
107E            ABUF:  DS      12      ;ASCII BUFFER
108A            BBUF:  DS      4       ;BINARY BUFFER
108E            SCNT:  DS      1
108F            DCNT:  DS      1       ;DUMP ROUTINE COUNTER
000B            NCOM   EQU     11      ;NUMBER OF COMMANDS
1090            TABA:  DS      2       ;SYMBOL TABLE END ADDRESS
1092            ASPC:  DS      2       ;ASSEMBLER PROGRAM COUNTER
1094            PASI:  DS      1       ;PASS INDICATOR
1095            NCHR:  DS      1       ;LENGTH OF STRING FOR COMPARE
1096            PNTR:  DS      2       ;LINE POINTER STORAGE
```

```
1098            NOLA:   DS      1       ;NUMBER OF LABELS
1099            SIGN:   DS      1       ;SIGN STORAGE FOR SCAN
109A            OPRD:   DS      2       ;OPERAND STORAGE
109C            OPRI:   DS      1       ;OPERAND FOUND INDICATOR
109D            TEMP:   DS      1
1072            APNT    EQU     INSP    ;ASSEMBLE LINE POINTER
108E            AERR    EQU     SCNT    ;ASSEMBLER ERROR PRINT SWITCH
109E            OIND:   DS      2       ;OUTPUT ADDRESS
0005            LLAB    EQU     5       ;LENGTH OF LABELS
10A0            AREA:   DS      18
10B2            OBUF:   DS      16      ;OUTPUT BUFFER AREA
10C2                    DS      5
10C7            IBUF:   DS      83
111A            SYMT    EQU     $       ;START OF SYMBOL TABLE
                ;
                ; TELETYPE PARAMETERS
                ;
0003            TTS     EQU     3       ;TTY STATUS PORT
0002            TTI     EQU     2       ;TTY DATA IN PORT
0002            TTO     EQU     2       ;TTY DATA OUT PORT
0002            TTYDA   EQU     2       ;TTY DATA AVAILABLE BIT
0001            TTYTR   EQU     1       ;TTY XMTR READY BIT
00FF            SWCH    EQU     0FFH    ;SWITCH REGISTER
                ;
0000                    END
```

ERRATA

REV. 1

Basic Key-In Instruction

```
00D4 B7                 ORA     A       ; CHECKSUM. IS IT ZERO?
```

Paper Tape Loader Instruction

```
                ;
FD31 DB03       LOOP2:  IN      3       ;GET SIO STATUS
```

Due to a printing error these instructions were run off the page.

AUG 02 1976

BOOTSTRAP LOADER

The IMSAI Bootstrap Loader is a system that allows the user to get a general paper tape loader into any region of RAM using only a 32-byte key-in. It requires an ASR33 teletype. To use this loader, proceed as follows:

1. Key in the basic bootstrap given below starting at location 0000.

   ```
   3E CE D3 03 3E 17 D3 03 21 20 00 06 F8 DB 03 E6
   02 CA 0D 00 DB 02 77 3C CA 08 00 23 05 C2 0D 00
   ```

2. Mount the bootstrap tape in the paper tape reader on the teletype so that the block of rubouts (frames with all the holes punched out) is in the reader.

3. Set the PROGRAMMED INPUT switches to the high order 8 bits of the address where the paper tape loader is to be located, e.g., to put the loader at 5C00 hex, set the PROGRAMMED INPUT switches to 5C hex. (See the warning below.)

4. Press STOP, RESET and RUN, then manually start the paper tape reader on the teletype.

If all goes well, the tape should go through the reader, stop at the end, then the loader will print an "*" on the teletype. If this is the case, refer to the IMSAI Paper Tape Loader section to use the loader.

If the loader does not type an asterisk after the tape has gone through the reader, this means the loader was not read in correctly. Proceed as follows:

1. Check the basic bootstrap key into it as correct.

2. If the key-in is correct, check the bootstrap tape for tears or distorted holes. (These may usually be fixed with cellophane tape.)

If the key-in and bootstrap tape are correct, the problem may be dirty contacts in the teletype reader. Try repeating the bootstrap procedure from the beginning.

WARNING:

1. Since the bootstrap loader resides in location 20 hex - 120, do not try to load the paper tape loader below 200 hex or it will overlay the bootstrap.

2. Be sure to locate the loader in a region where it will not be overlayed by the program it is loading. For instance, 8K BASIC occupies locations 0000-1FFF hex, so that to load 8K BASIC, the loader should be located at or above 2000 hex.

BOOTSTRAP LOADER PROGRAM LOGIC

The Bootstrap Loader is a system that allows the user to read the Paper Tape Loader into the region of RAM that begins on a 256-word boundary using a specially formatted tape.

1. Bootstrap Tape Format:

   The Bootstrap Tape consists of two sections. The first section consists of a direct core image of the second level bootstrap (described below), preceded by a block of rub-outs. In this section of the tape, each frame corresponds directly to one data byte. The second section consists of the Paper Tape Loader in standard object format.

2. Overall Logic:

   The Bootstrap Sequence Procedure is as follows:

   a. The user keys in a simple 32-byte bootstrap, starts it up, then starts the tape reader on the teletype.

   b. The basic bootstrap reads in the second level bootstrap from the first part of the bootstrap tape and starts it up.

   c. The second level bootstrap stops the tape reader then checksums itself to make sure it was loaded correctly. If not, it hangs up.

   d. If the second level bootstrap checksums correctly, it starts the tape reader and reads in the paper tape loader from the second part of the bootstrap tape and locates it in the 256-byte page specified by the PROGRAMMED INPUT switches. If it detects an error in the tape, it stops the reader and hangs up.

   e. When the Paper Tape Loader is completely loaded, it stops the paper tape reader, then starts up the Paper Tape Loader.

3. Basic Bootstrap:

   The Basic Key-In Bootstrap was designed to be as short as possible. It merely reads in characters from the tape and stores them directly into memory. Whenever it reads in a byte of FF hex, it resets its pointer and counter. This allows it to use the block of rubouts at the beginning of the tape to synchronize on.

4. Second Level Bootstrap:

The second level bootstrap is a modified version of the Paper Tape Loader. The main differences between the two are:

a. The second level bootstrap checksums itself to make sure it was loaded properly. This is done because the Basic key-in bootstrap, for reasons of brevity, does not error checking.

b. If it encounters an error, the second level bootstrap turns off the tape and hangs up.

c. If it encounters a byte of FD hex, it substitutes the contents of the PROGRAMMED INPUT switches. This is done so that the Paper Tape Loader may be located at any 256-byte page in memory. See below.

5. Relocating the Paper Tape Loader

The Paper Tape Loader that is on the second part of the bootstrap tape was assembled to begin at FD00 hex. Since there is no instruction with op-code FD hex, the only times a byte of FD hex will appear on the tape are:

a. The high byte of the address field in the paper tape record. (Note that the high byte of the address fields of all records will be FD hex.)

b. The high byte of the address in a jump instruction.

Therefore, by substituting another value (in this case, the contents of the PROGRAMMED INPUT switches) for every occurance of FD hex, we can load the Paper Tape Loader into any 256-byte page in memory.

PAPER TAPE LOADER

The IMSAI Paper Tape Loader is a program that will load tapes in the standard object format (see appendix) from the paper tape reader on an ASR33 teletype.

If the paper tape loader is read in with the bootstrap loader (see Bootstrap Loader section), it will start itself up and print an "*" on the teletype. Otherwise, it should be manually started at its beginning address.

When the loader prints an "*" on the teletype, mount the tape to be loaded in the paper tape reader on the teletype. Then, strike any key on the teletype. The paper tape reader should start automatically. While the tape is being read in, the data being loaded will be displayed in the PROGRAMMED OUTPUT lights.

The loader will stop the reader and print an "*" under two conditions:

1. If the PROGRAMMED OUTPUT displays 00 (all lights off), the loader has encountered an End-of-File record, an the program has been successfully loaded. At this point, another tape may be loaded by placing it in the paper tape reader and striking a key on the teletype.

2. If something other than 00 is displayed in the PROGRAMMED OUTPUT lights, a bad record has been encountered in the tape. The record may be re-read as follows:

   - Move the switch on the reader to the "FREE" position
   - Back the tape up about two feet
   - Put the switch back in the "STOP" position
   - Strike a key on the teletype

   If the loader stops again on the same record, inspect the tape for tears or distorted holes (these may usually be fixed with cellophane tape).

PAPER TAPE LOADER PROGRAM LOGIC

The IMSAI Paper Tape Loader is a program designed to load paper tapes in the standard object format from the paper tape reader on an ASR33 teletype. The loader is designed to use no stack or local RAM, thereby allowing it to be executed out of ROM.

1. Object Tape Format:

The standard object format is a blocked hexadecimal format. The data on the tape is blocked into discrete records, each record containing record length, record type, memory address and checksum information in addition to data. A frame-by-frame description is as follows:

Frame 0	Record Mark. Signals the start of a record. The ASCII character colon (":" 3A hex) is used as the record mark.
Frames 1,2 (0-9, A-F)	Record Length. Two ASCII characters representing a hexadecimal number in the range 0 to FF (0 to 255). This is the count of actual data bytes in the record type or checksum. A record length of 0 indicates end-of-file.
Frames 3 to 6	Load Address. Four ASCII characters that represent the initial memory location where the data following will be loaded. The first data byte is stored in the location pointed to by the load address; succeeding data bytes are loaded into ascending addresses.
Frames 7,8	Record Type. Two ASCII characters. Currently all records are type 0. This field is reserved for future expansion.
Frames 9 to 9+2*	Data. Each 8-bit memory word is represented by two frames containing the ASCII characters 0-9, A-F) to represent a hexadecimal value 0 to FF hex (0 to 255).
Frames 9+2* (Record Length) to 9+2* (Record Length + 1	Checksum. The checksum is the negative of the sum of all 8-bit bytes in the record since the record mark (":") evaluated modulus 256. That is, if you add together all the 8-bit bytes, ignoring all carries out of an 8-bit sum then add the checksum, the result is zero.

Example: If memory locations 1 through 3 contain 53F8EC, the format of the hex file produced when these locations are punched is:

:0300010053F8ECC5

2. Register Allocation:

Since this loader uses no RAM, all variables and data are kept in the registers. The registers are assigned as follows:

A - scratch
B - byte count for data field
C - checksum
D - holes the data byte
E - flag register, describes what to do next

If this register contains zero, this program is looking for a ":" to signal the beginning of a block. Otherwise, if bit 7=1, then the next character is the first digit of a byte. If bit 2=0, the next character is the second digit of a byte. Bits 0-6 have the following significance:

1 - next byte is a count
2 - next byte is a high byte of the load address
3 - next byte is a low byte of the load address
4 - next byte is a type byte
5 - next byte is a data byte
6 - next byte is a checksum byte.

H, L - Load Address.

3. Logic:

The program flow is controlled by the flags in the E-register as given above.

```
                        ;
                        ; *** BASIC KEY-IN BOOTSTRAP LOADER ***
                        ;
                        ; THIS SIMPLE LOADER BOOTSTRAPS IN THE SECOND
                        ; LEVEL BOOTSTRAP, WHICH IN TURN LOADS THE
                        ; REAL PAPER TAPE LOADER.
                        ;
                        ; TO USE THIS LOADER, PROCEED AS FOLLOWS:
                        ;       (1) KEY IN THIS LOADER, STARTING AT LOC 1000
                        ;       (2) MOUNT THE BOOTSTRAP TAPE, SO THAT
                        ;           THE BLOCK OF RUBOUTS AT THE BEGINNING
                        ;           OF THE TAPE IS IN THE READER
                        ;       (3) SET THE PROGRAMMED INPUT SWITCHES TO THE
                        ;           HIGH ORDER 8 BITS OF THE ADDRESS WHERE
                        ;           YOU WANT THE PAPER TAPE LOADER TO
                        ;           BE LOADED. (E.G. TO CAUSE THE LOADER
                        ;           TO BE LOADED AT 5C00, SET THE PROGRAMMED
                        ;           INPUT SWITCHES TO 5C.)
                        ;       (4) PRESS THE 'RESET' KEY, FOLLOWED BY THE
                        ;           'RUN' KEY, THEN MANUALLY START THE PAPER
                        ;           TAPE READER ON THE TELETYPE.
                        ;
                        ;  IF EVERYTHING GOES CORRECTLY, THE LOADER WILL STOP
                        ;  THE PAPER TAPE READER, AND PRINT A * ON THE
                        ;  TELETYPE.  AT THIS POINT, MOUNT THE TAPE TO BE
                        ;  LOADED IN THE TELETYPE READER, THEN STRIKE ANY KEY
                        ;  ON THE TELETYPE.  THE LOADER WILL START THE
                        ;  PAPER TAPE READER, AND START LOADING THE TAPE. IF
                        ;  IT FINDS ANYTHING WRONG WITH THE TAPE, IT WILL
                        ;  STOP THE READER.  LOADING MAY BE CONTINUED BY
                        ;  STRIKING A KEY ON THE TELETYPE.
                        ;
                        ;
00F8                    CNT     EQU     0F8H    ;SIZE OF 2ND LEVEL BOOTSTRAP
                        ;
0000 3ECE               BOOT1:  MVI     A,0CEH  ;GET MODE BYTE FOR SIO BOARD.
0002 D303                       OUT     03      ;ISSUE IT
0004 3E17                       MVI     A,17H   ;GET COMMAND BYTE
0006 D303                       OUT     03      ; ISSUE IT
0008 212000             B1RST:  LXI     H,B1END ;GET LOAD ADDRESS
000B 06F8                       MVI     B,CNT   ;GET # OF BYTES
                        ;
000D DB03               LOOP:   IN      03      ;GET STATUS
000F E602                       ANI     2       ;IS THERE A BYTE READY
0011 CA0D00                     JZ      LOOP    ;KEEP WAITING
0014 DB02                       IN      2       ;GET THE BYTE
0016 77                         MOV     M,A     ;STORE IT
0017 3C                         INR     A       ;WAS IT A RUBOUT?
0018 CA0800                     JZ      B1RST   ;IF YES, RESET POINTERS
001B 23                         INX     H       ;ELSE, BUMP POINTER
001C 05                         DCR     B       ;AND DECR COUNT
001D C20D00                     JNZ     LOOP    ;IF NOT DONE, GO GET ANOTHER
                                                ; CHAR. ELSE, FALL THROUGH AND
                                                ; START UP SECOND LEVEL
                                                ; BOOTSTRAP.
0020                    B1END   EQU     $
0000                            END
```

```
                        ;
                        ; SECOND LEVEL BOOTSTRAP
                        ;
                        ; THIS LOADER IS PULLED IN BY THE BASIC KEY-IN
                        ; LOADER. WHEN STARTED UP BY THE KEY-IN LOADER,
                        ; IT CHECKSUMS ITSELF, TO MAKE SURE THAT IT
                        ; HAS BEEN LOADED CORRECTLY, THEN PULLS IN AND
                        ; RELOCATES THE MAIN PAPERTAPE LOADER.
                        ;
                        ; NOTE THAT THIS LOADER IS A SLIGHTLY MODIFIED
                        ; VERSION OF THE MAIN PAPER TAPE LOADER.
                        ;
                        ;
0000                            ORG     20H
                        ;
0020 3E13       BOOT2:  MVI     A,13H   ;GET STOP CHAR
0022 D302               OUT     2       ;STOP THE READER
0024 06F7               MVI     B,CHKSM-BOOT2 ;GET SIZE OF LDR
0026 212000             LXI     H,BOOT2 ;GET ADDRESS OF LDR
0029 AF                 XRA     A       ;CLEAR A AND CARRY
                        ;
                        ; PERFORM AN END-AROUND CHECKSUM, TO MAKE SURE
                        ; WE WERE LOADED CORRECTLY.
                        ;
002A 8E         CHECK:  ADC     M       ;ADD IN A BYTE WITH CARRY
002B 23                 INX     H       ;BUMP POINTER
002C 05                 DCR     B       ;DECREMENT COUNT
002D C22A00             JNZ     CHECK   ;KEEP GOING
0030 CE00               ACI     0       ;ADD IN LAST CARRY
0032 BE                 CMP     M       ;COMPARE WITH CHECKSUM
0033 C23300     XXX:    JNZ     XXX     ;HANG UP IF NO GOOD.
                        ;
                        ; WE DO THE FOLLOWING NONSENSE BECAUSE THE
                        ; BASIC KEY-IN BOOTSTRAP WILL NOT LOAD
                        ; AN 0FFH CHARACTER.
                        ;
0036 218C00             LXI     H,FF1+1 ;GET ADDRESS OF 'IN 0FEH' INST
0039 34                 INR     M       ;MAKE IT 'IN 0FFH'.
003A 218100             LXI     H,FF2+1 ;DO IT AGAIN
003D 34                 INR     M
003E 210801             LXI     H,FF3+1 ;AND AGAIN
0041 34                 INR     M
                        ;
                        ; NOW WE'RE READY TO LOAD AND RELOCATE THE LOADER
                        ;
0042 C35E00             JMP     STR     ;1ST TIME, SKIP RE-INIT STUFF.
                        ;
0045 3EAA       START:  MVI     A,0AAH  ;GET DUMMY MODE BYTE
0047 D303               OUT     3
0049 3E40               MVI     A,40H   ;GET RESET COMMAND
004B D303               OUT     3       ;ISSUE IT
004D 3EFA               MVI     A,0FAH
004F D303               OUT     3       ;ISSUE MODE BYTE TO SIO
0051 3E17               MVI     A,17H
0053 D303               OUT     3       ;ISSUE COMMAND BYTE
0055 DB03       SL:     IN      03      ;GET STATUS
0057 E602               ANI     02      ;CHECK FOR CHAR READY
0059 CA5500             JZ      SL      ;KEEP WAITING
005C DB02               IN      02      ;READ CHAR AND IGNOR
005E DB03       STR:    IN      03      ;GET STATUS
0060 E601               ANI     1       ;MAKE SURE WE HAVE XMTR RDY
0062 CA5E00             JZ      STR
0065 3E11               MVI     A,11H   ;GET 'XON' CHAR
0067 D302               OUT     02      ;START READER
```

```
                        LOOP1:  MVI     E,0         ;CLEAR FLAG
006B 0E00                       MVI     C,0         ;CLEAR CHECKSUM
                        ;
006D DB03               LOOP2:  IN      3           ;GET SIO STATUS
006F E602                       ANI     2           ;CHECK FOR CHARACTER
0071 CA6D00                     JZ      LOOP2       ;KEEP WAITING
0074 7B                         MOV     A,E         ;GET FLAG
0075 B7                         ORA     A           ;IS IT ZERO?
0076 C28700                     JNZ     X1          ;NO, GO PROCESS A HEX CHAR
0079 DB02                       IN      2           ;YES, WE'RE LOOKING FOR A COLON
007B E67F                       ANI     127         ;STRIP OFF PARITY BIT
007D FE3A                       CPI     ':'         ;IS IT A COLON?
007F C26D00                     JNZ     LOOP2       ;NO, KEEP WAITING
0082 1E81                       MVI     E,81H       ;YES, SET FLAG FOR COUNT BYTE
0084 C36D00                     JMP     LOOP2       ;AND GET ANOTHER CHAR.
                        ;
                        ; WE'RE PUTTING TOGETHER A BYTE. FLAG BIT 7 = 1 => HIGH
                        ; DIGIT OF BYTE, BIT 7=0 => LOW DIGIT
                        ;
0087 F2A200             X1:     JP      Y1          ;JUMP IF LOW DIGIT
008A E67F                       ANI     127         ;ELSE STRIP OFF HIGH BIT
008C 5F                         MOV     E,A         ;PUT FLAG BACK IN E-REG
008D DB02                       IN      2           ;GET THE CHAR
008F E67F                       ANI     127         ;STRIP OFF THE PARITY BIT
0091 FE3A                       CPI     '9'+1       ;IS IT .LE. '9'
0093 FA9800                     JM      X2          ;SKIP IT YES
0096 C609                       ADI     9           ;IF NOT, ADJUST IT
0098 E60F               X2:     ANI     0FH         ;GET HEX DIGIT
009A 87                         ADD     A           ;SHIFT LEFT ONE BIT
009B 87                         ADD     A           ;     TWO BITS
009C 87                         ADD     A           ;     THREE BITS
009D 87                         ADD     A           ;AND FOUR BITS.
009E 57                         MOV     D,A         ;SAVE NIBBLE IN D REG
009F C36D00                     JMP     LOOP2
                        ;
                        ; PROCESS LOW DIGIT OF BYTE, THEN DECIDE WHAT TO DO WITH
                        ;
00A2 DB02               Y1:     IN      2           ;GET THE CHAR
00A4 E67F                       ANI     127         ;GET RID OF PARITY BIT
00A6 FE3A                       CPI     '9'+1       ;HEX IS SUCH A PAIN.
00A8 FAAD00                     JM      Y2
00AB C609                       ADI     9
00AD E60F               Y2:     ANI     0FH
00AF B2                         ORA     D           ;MAKE THE BYTE
00B0 D3FE               FF2:    OUT     0FEH        ;PUT IT IN LIGHTS
00B2 57                         MOV     D,A         ;SAVE IT IN D REG
00B3 81                         ADD     C           ;ADD IT INTO CHECKSUM
00B4 4F                         MOV     C,A         ;SAVE RUNNING CHECKSUM
00B5 7A                         MOV     A,D         ;GET BYTE BACK
00B6 FEFD                       CPI     0FDH        ;IS IT FELOCATABLE BYTE?
00B8 C2BD00                     JNZ     Y3          ;BRANCH IF NOT
00BB DBFE               FF1:    IN      0FEH        ;ELSE SUBSTITUE SWITCHS
00BD 57                 Y3:     MOV     D,A         ;PUT BYTE BACK IN D
00BE 7B                         MOV     A,E         ;GET FLAG IN A
00BF 3D                         DCR     A           ;THEN DISPATCH  ON IT
00C0 CA0401                     JZ      COUNT
00C3 3D                         DCR     A
00C4 CAFE00                     JZ      HADD
00C7 3D                         DCR     A
00C8 CAF800                     JZ      LADD
00CB 3D                         DCR     A
00CC CAF300                     JZ      TYPE
00CF 3D                         DCR     A
00D0 CAE700                     JZ      PUT
00D3 79                         MOV     A,C         ;MUST BE TIME TO CHECK THE
```

```
00D8 214500          LXI     H,START ;ELSE, GET RESTART ADDR
00DB 3E13      STOP: MVI     A,13H   ;GET 'XOFF' CHAR
00DD D302            OUT     2       ;TURN OFF READER
00DF DB03      STPL: IN      3       ;WAIT TILL XMTR BUFFER EMPTY
00E1 E604            ANI     4
00E3 CADF00          JZ      STPL
00E6 E9              PCHL            ;GO AWAY.
                     ;
                     ;
                     ; PUT A DATA BYTE INTO CORE
                     ;
00E7 72        PUT:  MOV     M,D     ;STORE THE DATA
00E8 23              INX     H       ;INCREMENT THE H REG
00E9 1E85            MVI     E,85H   ;RESET FLAG FOR NEXT DATA BYTE
00EB 05              DCR     B       ;DECR COUNT
00EC C26D00          JNZ     LOOP2   ;GO BACK FOR MORE DATA.
00EF 1C              INR     E       ;OUT OF DATA, SET FLAG FOR
00F0 C36D00          JMP     LOOP2   ;  CHECKSUM.
                     ;
                     ; IGNORE A TYPE BYTE
                     ;
00F3 1E85      TYPE: MVI     E,85H   ;SET FLAG FOR DATA
00F5 C36D00          JMP     LOOP2   ;GO GET DATA
                     ;
                     ; GET LOW BYTE OF ADDRESS
                     ;
00F8 6A        LADD: MOV     L,D     ;GET BYTE INTO L-REG
00F9 1E84            MVI     E,84H   ;SET FLAG FOR TYPE BYTE.
00FB C36D00          JMP     LOOP2
                     ;
                     ; GET HIGH BYTE OF ADDRESS
                     ;
00FE 62        HADD: MOV     H,D     ;GET BYTE INTO H
00FF 1E83            MVI     E,83H   ;SET FLAG FOR LOW ADDRESS BYTE
0101 C36D00          JMP     LOOP2
                     ;
                     ; GET COUNT BYTE
                     ;
0104 42        COUNT: MOV    B,D     ;PUT COUNT INTO B
0105 7A              MOV     A,D     ;CHECK FOR EOF
0106 B7              ORA     A
0107 C21201          JNZ     C1      ;IF NOT EOF, CONTINUE
010A DBFE      FF3:  IN      0FEH    ;GET HIGH BYTE OF LOADER
010C 67              MOV     H,A     ; ADDRESS INTO H
010D 2E00            MVI     L,0     ;AND LOW BYTE
010F C3D800          JMP     STOP    ;STOP TAPE, THEN GOTO LOADER.
                     ;
0112 1E82      C1:   MVI     E,82H   ;SET FLAG FOR ADDRESS BYTE
0114 C36D00          JMP     LOOP2
                     ;
                     ;
0117 C8        CHKSM: DB     0C8H    ;SELF-CHECKSUM FOR THIS LOADER
                     ;
0000                 END
```

```
                ;
                ; *** IMSAI PAPER TAPE LOADER ***
                ;
                ;       REV 0   3/3/76          A.WINGO
                ;
                ; THIS LOADER IS DESIGNED TO LOAD PAPER TAPES IN
                ; THE STANDARD OBJECT FORMAT (SEE THE SOFTWARE
                ; SECTION OF THE 8080 USER MANUAL) FROM AN ASR 33
                ; TELETYPE.  IT USES NO STACK AND NO LOCAL RAM, SO
                ; THAT IT MAY BE RUN FROM PROM WITHOUT REQUIRING
                ; A RAM CARD OF ITS OWN.
                ;
                ; USING THE LOADER:
                ;       IF THIS LOADER IS BROUGHT IN WITH THE
                ; BOOTSTRAP SEQUENCE (DOCUMENTED ELSEWHERE),
                ; IT WILL START ITSELF UP. OTHERWISE, MANUALLY
                ; START IT AT ITS BEGINNING.  IT WILL RESPOND
                ; BY TYPING A * ON THE TELETYPE. MOUNT THE TAPE
                ; TO BE LOADED IN THE READER, AND STRIKE ANY KEY.
                ; THE LOADER WILL START THE READER AUTOMATICALLY.
                ;       THE LOADER WILL STOP THE TAPE AND TYPE A * IN
                ; EITHER OF TWO CASES:
                ;
                ;       (1) IT HAS SEEN AN END OF FILE RECORD. IN
                ;           THIS CASE, ZERO WILL BE DISPLAYED IN
                ;           THE PROGRAMMED OUTPUT LIGHTS.
                ;
                ;       (2) IT ENCOUNTERED A BAD RECORD. IN THIS CASE
                ;           AN NON-ZERO QUANTITY WILL BE DISPLAYED
                ;           IN THE PROGRAMMED OUTPUT LIGHTS.
                ;
                ; IN EITHER CASE, LOADING MAY BE CONTINUED BY STRIKING
                ; A KEY.
                ;
                ;
0000                    ORG     0FD00H
                ;
FD00 110100     START:  LXI     D,1     ;WAIT ABOUT A SECOND SO A
FD03 210000             LXI     H,0     ;  PREVIOUS 'XOFF' CHARACTER
FD06 19         SL0:    DAD     D       ;  HAS TIME TO STOP THE READER
FD07 D206FD             JNC     SL0
                ;
                ; INITIALIZE SIO BOARD.
                ;
FD0A 3EAA               MVI     A,0AAH  ;GET DUMMY MODE BYTE
FD0C D303               OUT     3
FD0E 3E40               MVI     A,40H   ;GET RESET COMMAND
FD10 D303               OUT     3       ;ISSUE IT
FD12 3EFA               MVI     A,0FAH
FD14 D303               OUT     3       ;ISSUE MODE BYTE TO SIO
FD16 3E17               MVI     A,17H
FD18 D303               OUT     3       ;ISSUE COMMAND BYTE
FD1A 3E2A               MVI     A,'*'   ;GET AN ASTERISK
FD1C D302               OUT     02      ;PRINT IT
FD1E D802               IN      02      ;THROW AWAY ANY CHAR IN BUFFER
FD20 D803       SL2:    IN      03      ;GET STATUS
FD22 E602               ANI     02      ;CHECK FOR CHAR READY
FD24 CA20FD             JZ      SL2     ;KEEP WAITING
FD27 D802               IN      02      ;READ CHAR AND IGNOR
FD29 3E11               MVI     A,11H   ;GET 'XON' CHAR
FD2B D302               OUT     02      ;START READER
                ;
FD2D 1E00       LOOP1:  MVI     E,0     ;CLEAR FLAG
FD2F 0E00               MVI     C,0     ;CLEAR CHECKSUM
```

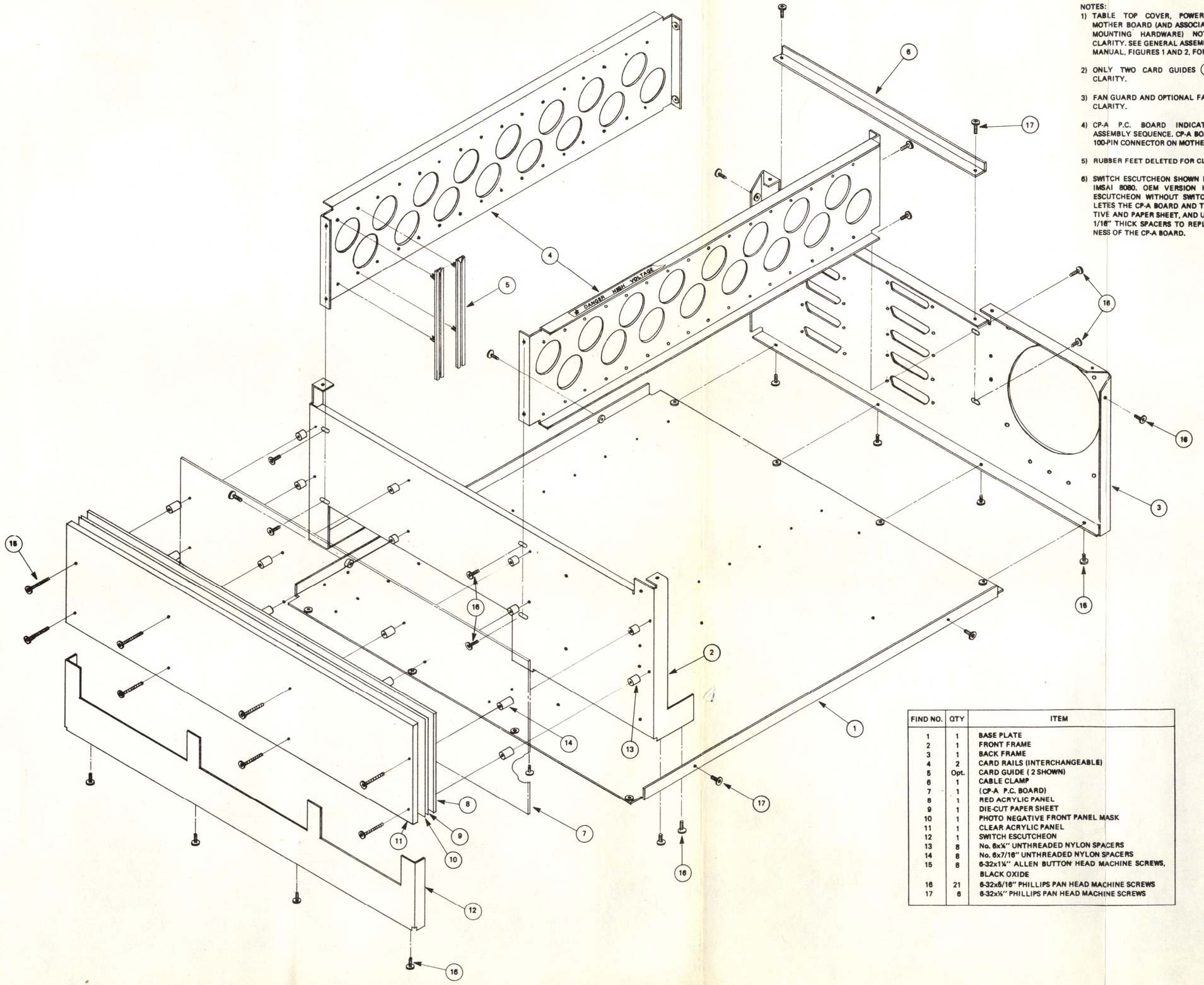
```
FD35 CA31FD            JZ      LOOP2     ;KEEP WAITING
FD38 7B                MOV     A,E       ;GET FLAG
FD39 B7                ORA     A         ;IS IT ZERO?
FD3A C248FD            JNZ     X1        ;NO, GO PROCESS A HEX CHAR
FD3D DB02              IN      2         ;YES, WE'RE LOOKING FOR A COLON
FD3F E67F              ANI     127       ;STRIP OFF PARITY BIT
FD41 FE3A              CPI     ':'       ;IS IT A COLON?
FD43 C231FD            JNZ     LOOP2     ;NO, KEEP WAITING
FD46 1E81              MVI     E,81H     ;YES, SET FLAG FOR COUNT BYTE
FD48 C331FD            JMP     LOOP2     ;AND GET ANOTHER CHAR.
                       ;
                       ; WE'RE PUTTING TOGETHER A BYTE. FLAG BIT 7 = 1 => HIGH
                       ; DIGIT OF BYTE, BIT 7=0 => LOW DIGIT
                       ;
FD48 F266FD    X1:     JP      Y1        ;JUMP IF LOW DIGIT
FD4E E67F              ANI     127       ;ELSE STRIP OFF HIGH BIT
FD50 5F                MOV     E,A       ;PUT FLAG BACK IN E-REG
FD51 DB02              IN      2         ;GET THE CHAR
FD53 E67F              ANI     127       ;STRIP OFF THE PARITY BIT
FD55 FE3A              CPI     '9'+1     ;IS IT .LE. '9'
FD57 FA5CFD            JM      X2        ;SKIP IT YES
FD5A C609              ADI     9         ;IF NOT, ADJUST IT
FD5C E60F      X2:     ANI     0FH       ;GET HEX DIGIT
FD5E 87                ADD     A         ;SHIFT LEFT ONE BIT
FD5F 87                ADD     A         ;     TWO BITS
FD60 87                ADD     A         ;     THREE BITS
FD61 87                ADD     A         ;AND FOUR BITS.
FD62 57                MOV     D,A       ;SAVE NIBBLE IN D REG
FD63 C331FD            JMP     LOOP2
                       ;
                       ; PROCESS LOW DIGIT OF BYTE, THEN DECIDE WHAT TO DO WITH
                       ;
FD66 DB02      Y1:     IN      2         ;GET THE CHAR
FD68 E67F              ANI     127       ;GET RID OF PARITY BIT
FD6A FE3A              CPI     '9'+1     ;HEX IS SUCH A PAIN.
FD6C FA71FD            JM      Y2
FD6F C609              ADI     9
FD71 E60F      Y2:     ANI     0FH
FD73 B2                ORA     D         ;MAKE THE BYTE
FD74 D3FF              OUT     0FFH      ;PUT IT IN LIGHTS
FD76 57                MOV     D,A       ;SAVE IT IN D REG
FD77 81                ADD     C         ;ADD IT INTO CHECKSUM
FD78 4F                MOV     C,A       ;SAVE RUNNING CHECKSUM
FD79 7B                MOV     A,E       ;GET FLAG IN A
FD7A 3D                DCR     A         ;THEN DISPATCH ON IT
FD7B CAC1FD            JZ      COUNT
FD7E 3D                DCR     A
FD7F CA88FD            JZ      HADD
FD82 3D                DCR     A
FD83 CA85FD            JZ      LADD
FD86 3D                DCR     A
FD87 CA80FD            JZ      TYPE
FD8A 3D                DCR     A
FD8B CAA4FD            JZ      PUT
FD8E 79                MOV     A,C       ;MUST BE TIME TO CHECK THE
FD8F B7                ORA     A         ; CHECKSUM. IS IT ZERO?
FD90 CA2DFD            JZ      LOOP1     ;YES, GO GET NEXT RECORD
FD93 2F        STOP:   CMA               ;DISPLAY REASON FOR STOPPING
FD94 D3FF              OUT     0FFH
FD96 3E13              MVI     A,13H     ;ELSE, GET 'XOFF' CHAR
FD98 D302              OUT     2         ;TURN OFF READER
FD9A DB03      STPL:   IN      3         ;WAIT TILL XMTR BUFFER EMPTY
FD9C E604              ANI     4
```

```
FDA1 C300FD              JMP     START
                    ;
                    ;
                    ; PUT A DATA BYTE INTO CORE
                    ;
FDA4 72             PUT:    MOV     M,D     ;STORE THE DATA
FDA5 23                     INX     H       ;INCREMENT THE H REG
FDA6 1E85                   MVI     E,85H   ;RESET FLAG FOR NEXT DATA BYTE
FDA8 05                     DCR     B       ;DECR COUNT
FDA9 C231FD                 JNZ     LOOP2   ;GO BACK FOR MORE DATA.
FDAC 1C                     INR     E       ;OUT OF DATA, SET FLAG FOR
FDAD C331FD                 JMP     LOOP2   ;  CHECKSUM.
                    ;
                    ; IGNORE A TYPE BYTE
                    ;
FDB0 1E85           TYPE:   MVI     E,85H   ;SET FLAG FOR DATA
FDB2 C331FD                 JMP     LOOP2   ;GO GET DATA
                    ;
                    ; GET LOW BYTE OF ADDRESS
                    ;
FDB5 6A             LADD:   MOV     L,D     ;GET BYTE INTO L-REG
FDB6 1E84                   MVI     E,84H   ;SET FLAG FOR TYPE BYTE
FDB8 C331FD                 JMP     LOOP2
                    ;
                    ; GET HIGH BYTE OF ADDRESS
                    ;
FDBB 62             HADD:   MOV     H,D     ;GET BYTE INTO H
FDBC 1E83                   MVI     E,83H   ;SET FLAG FOR LOW ADDRESS BYTE
FDBE C331FD                 JMP     LOOP2
                    ;
                    ; GET COUNT BYTE
                    ;
FDC1 42             COUNT:  MOV     B,D     ;PUT COUNT INTO B
FDC2 7A                     MOV     A,D     ;CHECK FOR EOF
FDC3 B7                     ORA     A
FDC4 CA93FD                 JZ      STOP    ;IF EOF, GO STOP READER
FDC7 1E82                   MVI     E,82H   ;ELSE SET FLAG FOR ADDRESS BYTE
FDC9 C331FD                 JMP     LOOP2
                    ;
                    ;
0000                        END
```

Schematic Drawings

APPENDIX — SCHEMATIC DIAGRAMS

CHASSIS CABINET
POWER SUPPLY
CPA
MPU-A
RAM 4A



NOTES:

1) TABLE TOP COVER, POWER SUPPLY, AND MOTHER BOARD (AND ASSOCIATED HOLES AND MOUNTING HARDWARE) NOT SHOWN FOR CLARITY. SEE GENERAL ASSEMBLY CHAPTER OF MANUAL, FIGURES 1 AND 2, FOR DETAILS.

2) ONLY TWO CARD GUIDES (5) SHOWN FOR CLARITY.

3) FAN GUARD AND OPTIONAL FAN DELETED FOR CLARITY.

4) CP-A P.C. BOARD INDICATED TO SHOW ASSEMBLY SEQUENCE. CP-A BOARD PLUGS INTO 100-PIN CONNECTOR ON MOTHER BOARD.

5) RUBBER FEET DELETED FOR CLARITY.

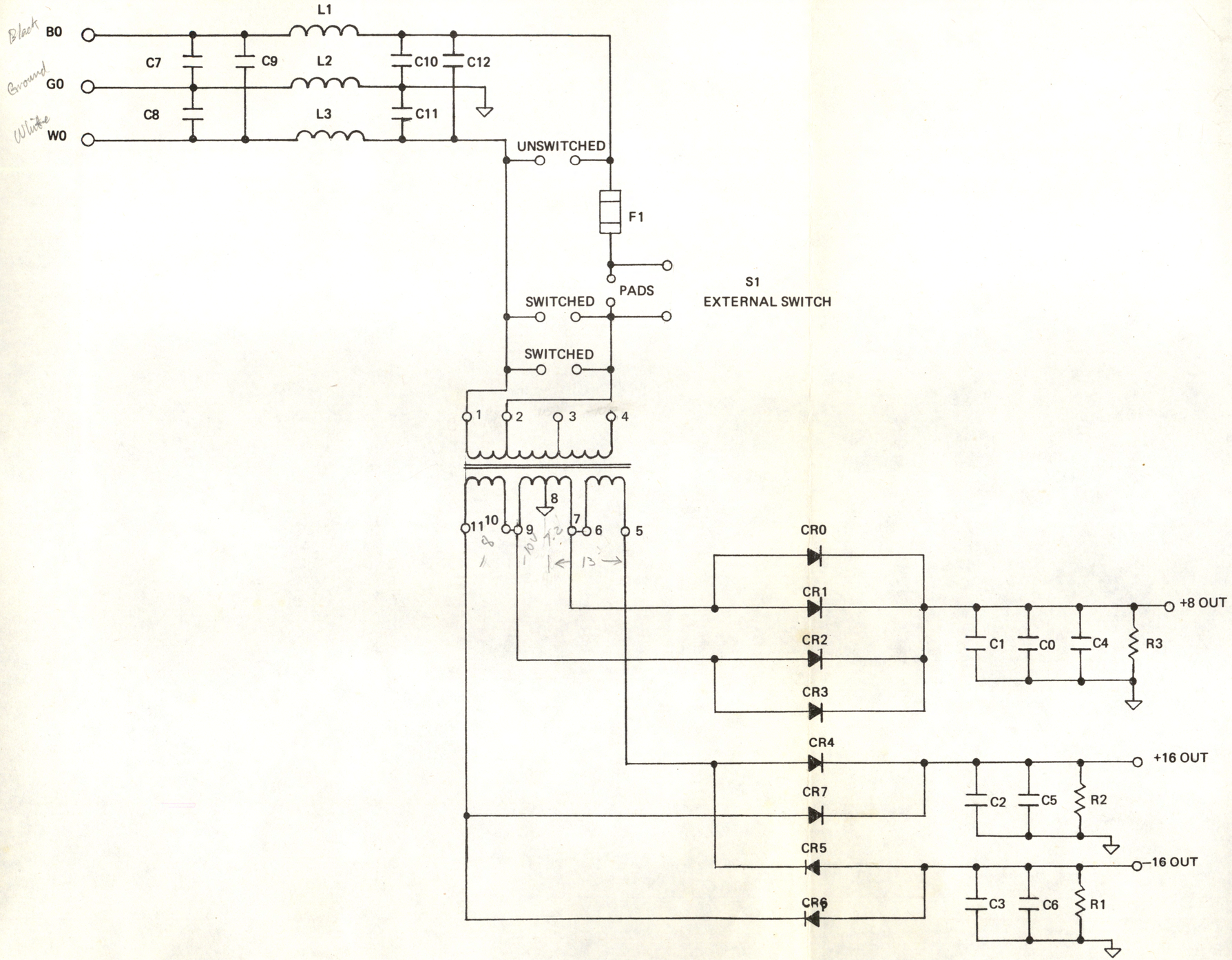
6) SWITCH ESCUTCHEON SHOWN IS FOR REGULAR IMSAI 8080. OEM VERSION HAS ALTERNATE ESCUTCHEON WITHOUT SWITCH CUTOUTS, DELETES THE CP-A BOARD AND THE PHOTO NEGATIVE AND PAPER SHEET, AND USES ADDITIONAL 1/16" THICK SPACERS TO REPLACE THE THICKNESS OF THE CP-A BOARD.

FIND NO.	QTY	ITEM
1	1	BASE PLATE
2	1	FRONT FRAME
3	1	BACK FRAME
4	2	CARD RAILS (INTERCHANGEABLE)
5	Opt.	CARD GUIDE (2 SHOWN)
6	1	CABLE CLAMP
7	1	(CP-A P.C. BOARD)
8	1	RED ACRYLIC PANEL
9	1	DIE-CUT PAPER SHEET
10	1	PHOTO NEGATIVE FRONT PANEL MASK
11	1	CLEAR ACRYLIC PANEL
12	1	SWITCH ESCUTCHEON
13	8	No. 6x¼" UNTHREADED NYLON SPACERS
14	8	No. 6x7/16" UNTHREADED NYLON SPACERS
15	8	6-32x1¼" ALLEN BUTTON HEAD MACHINE SCREWS, BLACK OXIDE
16	21	6-32x5/16" PHILLIPS PAN HEAD MACHINE SCREWS
17	6	6-32x½" PHILLIPS PAN HEAD MACHINE SCREWS

8080 CHASSIS CABINET ASSEMBLY EXPLODED VIEW

7/76

©1976
IMS ASSOCIATES, INC.
SAN LEANDRO, CA.



C0, C1	95K uF
C2, C3	10K uF
C4, C5, C6	.1 uF
C7 thru C12	.04 uF 500V
CR0 thru CR3	MR1121
CR4 thru CR7	MR501
F1	5A
L1 thru L3	8uH
R1, R2	1K ½W
R3	470 ½W
S1	NOT SUPPLIED

IMS ASSOCIATES INC.
SCHEMATIC DIAGRAM
PS-C REV. 1 5/76

©1976

Note: MRWRT IS
Disabled on front panel.
Pin 5 of U24 has been Removed
from socket.
6-10-81 SR

NOTE:
Use this Circuit
if another Manufactors
memory is use

PWR

DEPOSIT

SOUT

DATA ON

EXM NXT

EAI

EAO

EBI

EBO

POZ

MWRITE

SINP

NOOP

HAD

LAD

OADD

SSW DSB

PRESET

EXT CLR

RUN

XRDY

SINTA
SWO
SSTACK
SHLTA
SOUT
SM1
SINP
SMEMR

PINTE
PHLDA
PWAIT

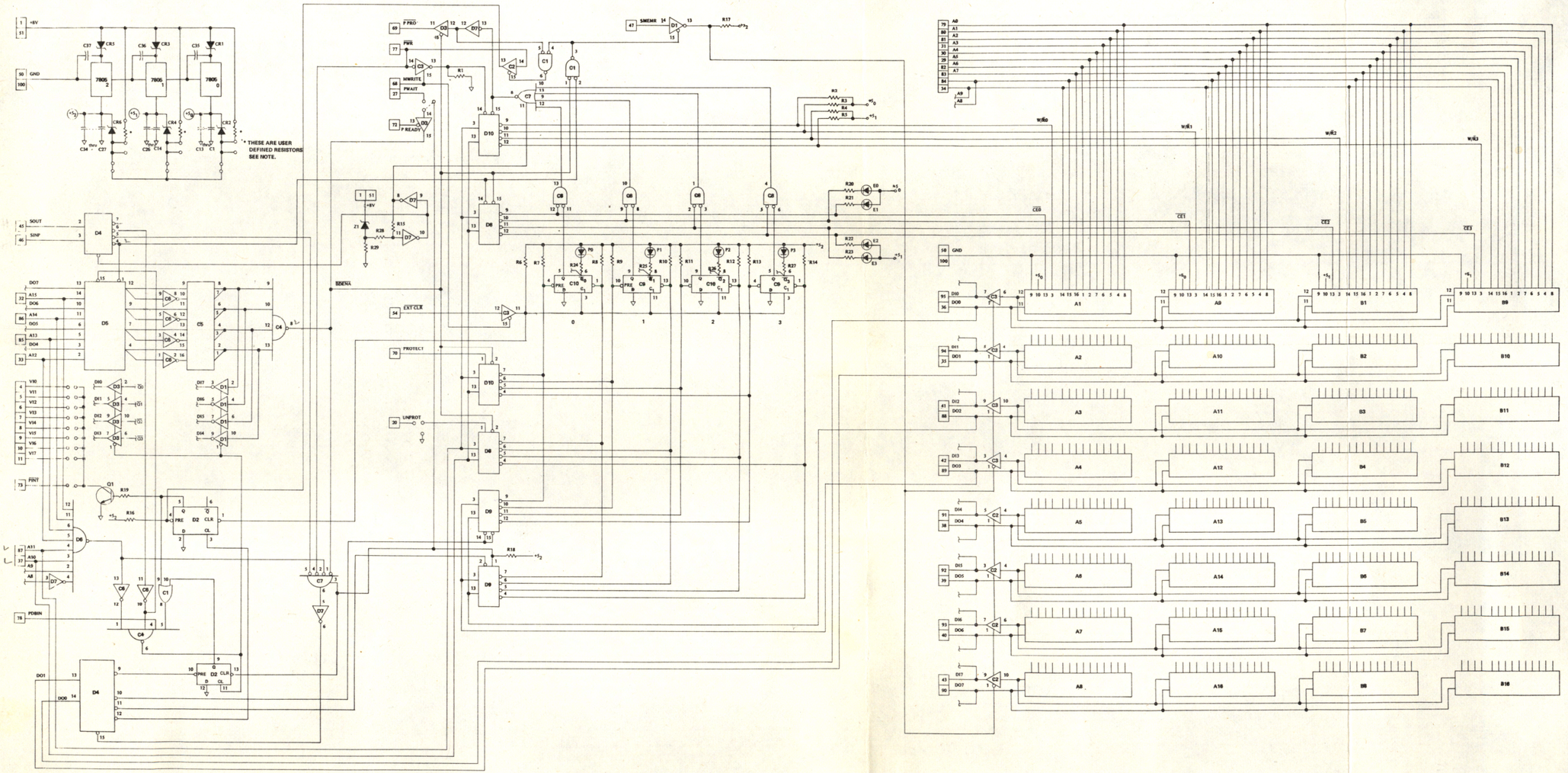
U1 U3 U4 U5 U7	7406
U2	DATA HEADER SOCKET
U6	8212
U8 U15	74LS04
U9	74LS30
U10	7404
U11	74LS10
U12	7410
U13 U16	7402
U14 U25	7400
U15.5	7427
U17 U20 U23	74123
U18 U19 U22	74107
U21	7430
U24	8T97
C1 C2 C5 C6 C7 C11 thru C22	.1mF
C3 C8	.001mF
C4	.01mF
C9 C10	33mF
CR1	1N914
LO0 thru LO7 LA0 thru LA15 LD0 thru LD7 LS0 thru LS7 LIE LHT LRN LWT	LED'S
R1	270K¼W
R2 R6 R7 R10 R11 R13 thru R15 R60	1K¼W
R3 R4 R5 R8 R9 R12	47K¼W
R16 thru R59	220 ¼W

IMS Associates Inc.
SCHEMATIC DIAGRAM
CPA REV 4 2/76

2/27/76

FRONT CONTROL PANEL

CPA
FAIRFIELD CORP HANGER EFH15B



A1 thru A16	2102
B1 thru B16	2102
C1	7432
C2 C3	8T97
C4	74LS20
C5	16 PIN JUMPER SOCKET
C6	7404
C7	7425
C8	7402
C9 C10	74LS74
D1	8T98
D2	74LS74
D3	8T97
D4	74LS139
D5	74LS157
D6	7430
D7	7404
D8 D9 D10	74LS156
C1 thru C34	.1 uF
C35 thru C37	33 uF
CR1 thru CR4	1N4002
Z1	1N751A
Q1	2N3904
R1 thru R18	1K ¼ WATT
R19 thru R28	220 ¼ WATT
R29	100 ¼ WATT
E0 thru E3	GREEN LED
P0 thru P3	RED LED

NOTE *:
THESE ARE USER DEFINED RESISTORS. REFER TO USERS GUIDE FOR EXPLANATION.

IMS ASSOCIATES INC.

SCHEMATIC DIAGRAM

RAM 4A--4 REV. 2 5/76

©1976